RANDOMIZATION AND COMPUTATION IN STRATEGIC SETTINGS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Shaddin Dughmi

August 2011

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Tim Roughgarden) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Serge Plotkin)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Amin Saberi)

Approved for the University Committee on Graduate Studies

Abstract

This thesis considers the following question: In large-scale systems involving many self-interested participants, how can we effectively allocate scarce resources among competing interests despite strategic behavior by the participants, as well as the limited computational power of the system? Work at the interface between computer science and economics has revealed a fundamental tension between the economic objective, that of achieving the goals of the system designer despite strategic behavior, and the computational objective, that of implementing aspects of the system efficiently. In particular, this tension has been most apparent in systems that allocate resources deterministically.

The realization that careful use of randomization can reconcile economic and computational goals is the starting point for this thesis. Our contributions are twofold: (1) We design randomized mechanisms for several fundamental problems of resource allocation; our mechanisms perform well even in the presence of strategic behavior, and can be implemented efficiently. (2) En route to our results, we develop new and flexible techniques for exploiting the power of randomization in the design of computationally-efficient mechanisms for resource allocation in strategic settings.

Acknowledgements

I owe a debt of gratitude to the many remarkable people I have had the fortune of knowing both during and before grad school, and without whom neither this thesis nor the memorable experiences of the last five years would have been possible.

I begin by thanking those who have influenced and supported my growth as an academic during my time at Stanford. First and foremost among them is my adviser, Tim Roughgarden. Tim's carefully-tuned mentorship and support were instrumental to my academic achievements. He always knew the right way to think about a research direction, the right way to explain an idea in a talk, and the right way to tackle a professional challenge, technical or otherwise. I also learned a lot from Tim through his example; his work ethic and clarity of thought have bolstered my own.

There are many others who's collaboration and/or company has influenced me academically during grad school. I thank Shahar Dobzinski for introducing me to many of the questions in this area, and for being a brilliant collaborator with a natural instinct for the right way to approach a problem. I thank Bobby Kleinberg for being a joy to work with; I found his enthusiasm infectious and his mathematical wizardry inspirational. I thank Mukund Sundararajan for his sage advice on all matters research and ping pong related. I thank Mike Hamburg for his time spent chasing down my reckless mathematical conjectures, teaching me a lot in the process. I thank Qiqi Yan for being great to work and share an office with. I thank Jan Vondrák for teaching me all about submodularity during my quals and beyond, and for his brilliant insights in our work together. I thank Jason Hartline for collaborating with me on beach volleyball in the guise of doing research, and for being a valuable resource on all matters auction theory related. I thank Serge Plotkin for the foundation in

approximation algorithms I gained through taking his courses, and for serving on my quals, defense, and thesis committees. I thank Ashish Goel, Ramesh Johari, and Amin Saberi for serving on my thesis or defense committee. Of those not already mentioned, I thank the following colleagues for their collaboration, insights, or otherwise shaping my research, in alphabetical order: Ittai Abraham, Ioannis Antonellis, Moshe Babiaoff, Dave Buchfuhrer, Liad Blumrosen, Anish Das Sarma, Peerapong Dhangwatnotai, Arpita Ghosh, Amos Fiat, Hu Fu, Brendan Lucier, Ankur Moitra, Noam Nisan, Ian Post, and Alex Slivkins. I also thank everyone else in the Stanford theory group and Gates 4B wing who made coming in to the office worth it.

My rewarding journey in theoretical research had its roots long before grad school, and I thank some of the people without whom I would have not found my way to this PhD. My introduction to the formal study of algorithms came at the hands of Jon Kleinberg and Éva Tardos, who's introductory class at Cornell sparked a sense of wonder of algorithms that is with me to this day. My first research experience was under the guidance of John Hopcroft, and I thank him for his mentorship and support. Then there's Joshua Guttman, who's impact on my growth as a researcher and a thinker can not be overstated; Joshua taught me how to see the big picture behind technical details, and how to reason clearly and speak thoughtfully.

I also had the fortune of befriending many remarkable people in the last five years, who's support and companionship made my time here as enjoyable as it was: my incoming class of computer science grad students, my fellow Rains community associates (CAs), fellow beach volleyball enthusiasts, and others. As my long-time co-CA and cherished friend, Sara Brownell deserves special thanks for all the laughs, epic adventures, and deficit neighborhood spending. I thank my incoming cohort of fellow CS grad students for their steady friendship, in particular Phil Guo, Mike Hamburg, Imran Haque, Robert Ikeda, Ewen Cheslack-Postava, and Chris Brigham. Other friends from Rains and/or the volleyball courts who deserve thanks for all the good times are: Shandor Dektor, Justine Kimball, Sumona Nag, Christian Perez, Asha Smith, Kelly Hennigan, Praveen Bommannavar, Sara Houts, all the Rains CAs, and my intramural volleyball teammates. I also thank Laura Holland for her support during the rough-and-tumble final year of grad school.

I owe special thanks to my family for instilling in me a love of learning and a desire to excel, and for being unwavering in their support of my goals. In particular I thank my mother, Reema Safadi, for her support and sacrifice through the years.

Finally, this work would not have been possible without the support of some fine institutions. I thank Stanford University for creating a idyllic paradise for graduate student life in all its respects, be they academic, social, or otherwise. I thank Cornell University for fond memories, and for a fine education grounded in the fundamentals. I also thank the the National Science Foundation and the Siebel Foundation for their support of this research.

Contents

List of Tables

List of Figures

List of Algorithms

Part I

Background and Overview

Chapter 1

Introduction

1.1 This Thesis in a Nutshell

This thesis considers a fundamental question at the interface between computer science and economics: In large-scale systems involving many self-interested participants, how can we effectively allocate scarce resources among competing interests despite strategic behavior by the participants, as well as the limited computational power of the system? This is the mission of *algorithmic mechanism design*, a field born of a collision between strategic behavior in economic environments on one hand, and the realities of computational intractability in large systems on the other.

Variants of this basic question are increasingly encountered in many different contexts: governments the world over are faced with the task of dividing the electromagnetic spectrum among competing telecommunications companies so as to maximize utilization of the spectrum; Internet Service Providers seek to divide limited bandwidth among their end users in order to maximize customer satisfaction; and content distribution companies must build overlay networks that are limited in size and yet adequately serve the distribution needs of their customers. In these examples, among others, a central agency must decide how to allocate resources among competing uses in order to achieve a stated global objective. This is complicated by the fact that many self-interested participants have a stake in the outcome, and moreover the set of possible allocations of resources can be complex and intractable to explore.

We distill two main challenges facing the designer of a system for allocating resources: one *economic* in nature, and the other *computational*. The economic challenge arises in situations where arriving at a "good" allocation of resources — usually defined as one that serves the interests of the participants in some aggregate sense — requires some cooperation from the participants in the system; say, through honestly reporting how "desirable" the participant finds each allocation. Unless a participant is properly *incentivized*, for example via the promise of a desirable allocation for the participant, a monetary payment, or more commonly a combination of the two, he will manipulate the system for his best-interest at the expense of the whole through misreporting this privately held information. On the other hand, the computational challenge is due to the realities of computational complexity; even if participants truthfully revealed their private information with no strings attached, adequately processing this information and aggregating it into the "best" global allocation of resources can be a computationally daunting task. In fact, adopting *polynomial time*[1] as the standard for computational efficiency, and assuming widely-held complexity-theoretic conjectures such as $P \neq NP$, efficient computation of the best global allocation of resources is provably impossible in many contexts.

Both challenges have been extensively studied in isolation. The economic field of *mechanism design* is concerned with designing protocols, known as *mechanisms*, whereby a central agency, which we refer to as the *principal*, communicates with other participants in the system, referred to as the *players*, and decides on an allocation of resources and a monetary payment from each player. We desire mechanisms that are *incentive compatible*, in that self-interested behavior by the players in the mechanism yields a globally desirable allocation of resources.[2] On the other hand, computer scientists have designed *approximation algorithms*. Given the correct private data of various players, these polynomial-time algorithms circumvent intractability by

[1] An algorithm runs in polynomial time if it terminates after a number of steps that is bounded by a polynomial in the length of its input. This is the standard notion of computational efficiency, and is defined formally in Chapter 2.

[2] This thesis designs mechanisms that satisfy a strong notion of incentive compatibility: each participant maximizes his expected payoff by truthfully reporting his private information to the mechanism, regardless of the actions of other participants. These mechanisms are called *truthful in expectation*, and are defined formally in Chapter 2.

computing *near-optimal* allocations of resources, rather than insisting on the "best" allocation. Both mechanism design and approximation algorithms have spawned a rich theory in the latter part of the twentieth century, with general and robust techniques that enable the design of useful mechanisms and approximation algorithms, respectively, for many central problems of resource allocation.

The fields of mechanism design and approximation algorithms collided at the turn of the century when researchers observed that, for many resource allocation problems encountered in theory and practice, standard approximation algorithms can not be used as part of a mechanism that performs well in the presence of strategic behavior, and standard economic mechanisms can not be implemented in polynomial time. This realization precipitated the birth of *algorithmic mechanism design*, the central question of which can be stated as follows:

> *To what extent is* incentive-compatible *and* polynomial-time *computation less powerful — in terms of approximating fundamental resource allocation problems — than "classical" polynomial-time computation?*

In the decade that followed, a large body of work has piled on evidence of a fundamental tension between the economic goal of incentive compatibility and the computational goal of polynomial time. This evidence manifested in the form of impossibility results for specific resource allocation problems, the strongest of which can be paraphrased as follows: despite the existence of both an incentive-compatible mechanism and a polynomial-time algorithm that each compute a near-optimal allocation of resources, an incentive-compatible *and* polynomial-time mechanism must output an allocation that is far from optimal on some inputs. While these impossibility results lend support to the thesis that incentive compatibility and polynomial-time computation are at loggerheads, *almost all of them are constrained to deterministic mechanisms.*

The realization that careful use of randomization can reconcile economic and computational goals is the starting point for this thesis. Our contributions are twofold: (1) We design randomized mechanisms for a variety of fundamental resource allocation problems that are incentive compatible, run in polynomial time, and match

the approximation guarantee of the best (non-incentive-compatible) polynomial-time algorithm. Our approximation guarantees are the best possible, given standard complexity-theoretic assumptions such as $P \neq NP$. Moreover, for most problems we consider, we improve on the best previously-known approximation guarantee of an incentive-compatible and polynomial-time mechanism by a constant or super-constant factor. (2) En route to our results, we develop new techniques for effectively using randomization to combine incentive compatibility and computational efficiency. Each of our techniques is applied to different resource allocation problems, and holds promise for even more general application. Moreover, one of our techniques yields computationally-efficient and incentive-compatible mechanisms that obtain the best-possible approximation guarantee for a large and natural class of problems that is abstractly defined.

Road map. The remainder of this introduction is organized as follows. First, Section 1.2 presents three concrete problems that illustrate the economic and computational challenges in resource allocation, and motivate the considerations of this thesis. These examples will serve as footholds during this introduction, and we reference them throughout to make our discussion more accessible. Section 1.3 informally presents our model and design goals, and introduces relevant terminology to facilitate our discussion. Section 1.4 describes, at a high level, the tension between incentive compatibility and computational efficiency, and Section 1.5 presents our intuition as to how randomization alleviates this tension. Section 1.6 informally describes our contributions, and Section 1.7 discusses related work. Finally, Section 1.8 gives some tips for reading the remainder of this thesis, and Section 1.9 closes this introduction with some bibliographic notes.

1.2 Three Illustrative Examples

To motivate the questions considered in this thesis, as well as to illustrate some of the basic concepts, we present three natural resource allocation problems, and examine each from both the economic and computational perspectives. The first, the problem

of allocating a single item, illustrates the essential principles of mechanism design, yet is computationally uninteresting. The second and third examples, combinatorial allocation and knapsack allocation, are representative of the complex resource allocation problems treated in this thesis.

1.2.1 Single Item Allocation

Consider a principal interested in auctioning a single item. He is approached by several potential buyers, the *players*. Each player has a certain *value* for the item, measured in monetary units. We consider a principal who wishes to maximize *social welfare* – the total value created in society – by awarding the item to the player who values it most.

If players' values for the item were known to the principal, his task would be simple. In reality, however, the difficulty is twofold: First, a player's value for the item is a function of his *private* circumstances and preferences, which are typically unknown to the principal. Second, players may misrepresent their values *strategically* if asked to provide information, particularly if they anticipate that doing so increases their likelihood of obtaining the item. This presents an informational constraint: how should a principal simultaneously learn the the players' values and at the same time use this information in his decision?

This question was explored by Vickrey [85], who described the following auction: solicit a *bid* from each player, then give the item to the highest bidder, and finally charge the winner of the auction a payment equal to the *second highest* bid. This familiar protocol, known commonly as the *Vickrey auction* or the *second-price auction*, is similar in essence to auctions employed on eBay, and emulates in one step the "open outcry" *English auction*.

Presumably, a player prefers to win the auction so long as his payment is less than his value for the item, and otherwise he prefers to lose. Under this natural assumption, the Vickrey auction, through its careful choice of payments, is *incentive compatible* in the following strong sense: a player can never gain by reporting a bid different from his true value, regardless of the actions of other players. This is evident

by a simple case analysis: Consider a player with value v for the item, and let p denote the maximum bid of all other players. Observe that the player wins the item at price p if he bids more than p, and loses and pays nothing if he bids less than p. If $p < v$, the player prefers the former outcome, and if $p > v$ he prefers the latter. In both cases, bidding his true value v realizes his preferred outcome.[3]

Because of incentive compatibility, "rational" players bid their values when faced with the Vickrey auction, and the item is awarded to the player who values it most. The Vickrey auction is an incentive-compatible and welfare-maximizing mechanism for the problem of allocating a single item. The same cannot be said of the *first-price auction*, which charges the highest bidder his own bid. Specifically, the winning player in the first-price auction frequently has incentive to "shade" his bid below his true value. The insights behind the Vickrey auction laid the foundations for mechanism design theory.

1.2.2 Combinatorial Allocation with Coverage Valuations

Definition

In practical settings more complex than single-item allocation, a principal has multiple items for sale. Since some items act as substitutes to each other, while others complement each other, a potential buyer's value must be described as a function of the *package* of items he receives. Auctions that assign items while taking into account such dependencies are known as *Combinatorial Auctions*. Such auctions are at the center of much of modern mechanism design theory, and they have been applied or considered in many contexts, such radio spectrum allocation, scheduling take-off and landing slots at airports, bandwidth allocation, and more. Due to the abstract nature of the general combinatorial allocation problem, which is described in Chapter 5, we present in this section a concrete special case.

In our example the principal is a government body that is tasked with dividing broadcasting rights for the radio spectrum among competing telecommunications

[3]This analysis easily extends to scenarios where $p = v$, because the player is indifferent between winning and losing in that case.

companies, the *players*. These rights are presented as *licenses*, the *items* in our example, where each license gives its owner the right to broadcast at a particular range of frequencies in a particular geographic region. The government must *allocate* the licenses among the players, a task commonly accomplished by employing a specialized *auction*. The players compete for these licenses, with the goal of generating revenue through providing broadcast services — such as, say, mobile telephone service — to potential customers. The potential revenue of player depends of the package of licenses it receives, and we refer to it as the player's *value* for that package. We consider a government interested in maximizing *social welfare*, defined as the total value of the players for the licenses they receive.

We simplify our problem further. We assume that the map from packages of licenses to a player's value, commonly referred to as the player's *valuation function*, is of a particular form that is illustrated in Figure 1.1. Specifically, a player associates each license with a set of potential customers: those living in the license's geographical region, capable of "listening" on the associated frequencies, and interested in the player's services. Moreover, the player associates a monetary amount with each such potential customer — or, more realistically, with a demographic group — measuring anticipated revenue. Naturally, a player's value for a package of licenses is the sum, over all customers "covered" by these licenses, of the customer's associated revenue. We call valuations of the form just described *coverage valuations*. We note that a customer may be covered by multiple licenses[4], and therefore the marginal value to a player of adding an additional license to his package decreases as his package of licenses grows.

Incentives

The set of customers that a player may service with a particular license, as well as the anticipated associated revenue, depends on factors specific to the player, such as their technical capabilities and position in the market. It is therefore natural to presume that the description of a player's valuation function — a "coverage pattern"

[4]Consider, for example, a potential customer with a multi-band phone, capable of receiving phone service on multiple frequency bands.

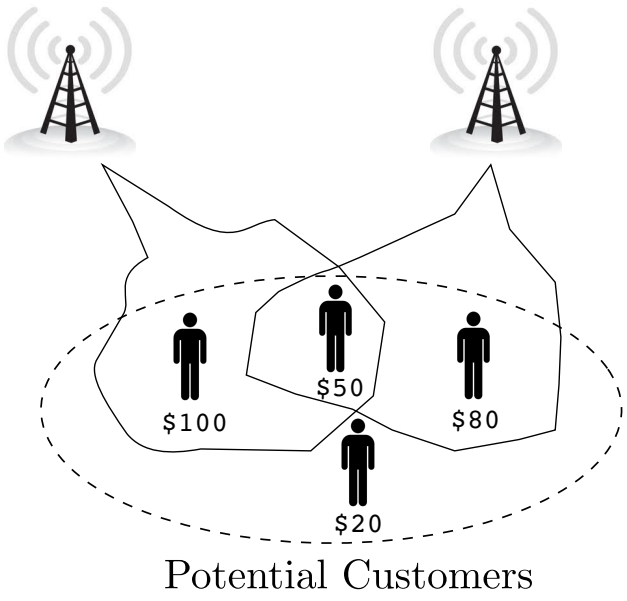


Figure 1.1: A coverage valuation with two licenses.
Each license is depicted as a radio tower. The player's value for the license on the left alone is $150. His value for the license on the right alone is $130. His value for the package of both licenses is $230.

as in Figure 1.1 — is private to the player. Moreover, we assume that a player behaves strategically in any mechanism in order to maximize his *utility*, defined naturally as his value for the set of licenses he receives less any monetary payment he is charged. This is the source of the economic challenge: A protocol for computing a welfare-maximizing allocation of the licenses must learn each player's private valuation function, whereas a self-interested player would misreport his valuation function if doing so improves his utility.

In the absence of computational limitations, classic work in mechanism design theory shows the existence of an incentive-compatible mechanism that computes a welfare-maximizing allocation of licenses. This mechanism is a generalization of the Vickrey Auction (Section 1.2.1) due to Clarke [21] and Groves [46], and is thus known as the *Vickrey-Clarke-Groves (VCG)* mechanism. The VCG mechanism solicits players' valuation functions, then finds a welfare-maximizing partition of the licenses among the players based on the reported valuations, and finally charges each player

carefully crafted payments. Concretely, each player i's payment in VCG is his *externality*, defined as the increase in the welfare of players other than i, based on their reported valuations, in the hypothetical scenario where player i leaves the auction and welfare is maximized for all remaining players. These payments essentially "internalize the externality", effectively "aligning" a player's goal of maximizing his utility with the principal's goal of maximizing the social welfare. This leads to incentive compatibility. The VCG mechanism applies in settings much more general than combinatorial allocation, and is described more formally in Chapter 3.

Computation

We can isolate computational properties of our special case of the combinatorial allocation problem by assuming that each player's true valuation function is presented up-front, as a coverage pattern as in Figure 1.1. We consider an algorithm for computing an allocation computationally efficient if it runs in time polynomial in the number of bits used to represent the players' valuation functions.

It is known that there is no polynomial-time algorithm that always outputs a welfare-maximizing allocation of licenses unless $P = NP$. In other words, the problem of welfare maximization in combinatorial allocation with coverage valuations is NP-hard. Computer scientists have traditionally coped with intractability by designing efficient *approximation algorithms*. The best known polynomial-time approximation algorithm for the combinatorial allocation problem with coverage valuations guarantees 63% of the maximum possible social welfare [87], and no polynomial-time algorithm provides a superior approximation guarantee unless $P = NP$ [54].

We remark that restricting our discussion to coverage valuations was in-part motivated by their computational properties. Combinatorial allocation with coverage valuations admits "useful" polynomial-time algorithms, in particular ones that guarantee a constant fraction of the maximum social welfare possible, independent of the number of players or the number of items. Whereas other classes of *restricted valuations* share this desirable property, the same cannot be said for *unrestricted valuations*.

1.2.3 Knapsack Allocation

Definition

Our next example is a strategic variant of the classical knapsack problem. Consider a principal tasked with implementing some subset of a collection of potential *projects*. Each project is associated with a *cost*, and the principal is constrained by a known *budget*. We say a set of projects is *feasible* if its total cost does not exceed the budget. There are several *players* who derive utility from projects to various extents. Specifically, each player associates a *value* with each project, and a player's value for a set of projects is defined as the sum over his values for the individual projects in that set. The goal of the principal is to implement a feasible set of projects maximizing the *welfare* of the players, defined as the sum of their values for the implemented set of projects.

Variants of this problem occur naturally in many contexts. In one variant, the principal is a government or municipality, the projects a set of government programs, and the players representatives of various constituencies with divergent priorities regarding how public resources are used. Another example appears regularly in online display advertising. Here, the principal is the content provider, and the players are interested advertisers. Each advertiser comes with a set of online banners of various sizes, and the budget is the total advertising space available on the content provider's homepage. In these examples, among others, it is natural to assume that costs of the projects and the principal's budget are publicly known, and we make this assumption here. We explore the knapsack allocation problem more formally in Chapter 8.

Incentives

We make similar assumptions to those made in Section 1.2.2. Specifically, it is natural to assume that a player's value for a project is *private*, and moreover the player acts to maximize his *utility* in any mechanism, defined as his value for the set of implemented projects less any payment he makes.

As in Section 1.2.2, the principal may apply the Vickrey-Clarke-Groves mechanism to this problem. Concretely, VCG asks each player to report his value for each

potential project, then computes the feasible set of projects maximizing the welfare of the players, and finally charges each player his externality. The VCG mechanism is incentive compatible in this context as well: players maximize their utility by reporting their true values to the mechanism.

Computation

In the absence of incentive considerations, this problem is simply the classic knapsack problem, studied extensively in computer science and operations research. It is known that the knapsack problem is NP-hard. Fortunately, however, it admits what can be thought of as the best possible approximation algorithm for an NP-hard problem, a *Fully Polynomial Time Approximation Scheme (FPTAS)*. An FPTAS for the knapsack problem is an algorithm that takes as input the costs, budget, and (true) values, as well as a parameter ϵ, executes in time polynomial in the length of the description of the input and $\frac{1}{\epsilon}$, and outputs a feasible set of projects with welfare at least a $(1-\epsilon)$ fraction of the best possible subject to feasibility.

1.3 Informal Preliminaries

We now informally define our model and introduce some relevant terminology, referencing the examples of Section 1.2 for concreteness. Concepts presented in this section are treated more formally in Chapters 2 and 3.

1.3.1 Mechanism Design Optimization Problems

A *mechanism design problem* is given by a set of possible *allocations*, also known as *outcomes* or *feasible solutions*, and a set of *players*, each equipped with a private *valuation function* from allocations to real numbers. A valuation function maps an allocation to the player's value for it, measured in monetary units. In the single item allocation problem with n players, the allocations correspond to the n possible choices of a player to whom the item is awarded,[5] and the valuation function of player

[5]Frequently, the allocation where no player receives the item is also considered feasible.

i, described by a single real number v_i, maps the allocation where player i receives the item to v_i, and other allocations to 0.

We consider mechanism design problems equipped with an *objective function*: a function that measures the "quality" of an allocation. Many objective functions are possible, and most natural ones aggregate the preferences of players (and/or the principal) in some way. The objective we consider in most of this thesis is *social welfare*: the sum of the players' values for the allocation. The problems in Section 1.2 were all posed with a welfare objective, though variants of these problems with other objectives have also been studied. We consider some non-welfare objectives in Chapter 10.

Finally, we distinguish mechanism design problems where player valuations are described by a single private parameter, known as *single-parameter* problems, from *multi-parameter* problems. We defer formal definition of this distinction to Chapter 10. For concreteness, however, we mention that the single-item allocation problem is single parameter, because each player's value for the item is a single real number, whereas the combinatorial allocation problem is multi-parameter because a player's valuation is described by a different real number for each package of the items. This distinction is important because incentive compatibility is typically much more permissive for single-parameter problems than it is for their multi-parameter counterparts, which enables better positive results in some contexts.

1.3.2 Mechanisms

We design *direct-revelation mechanisms* for mechanism design problems. These protocols consist of three steps: (1) Solicit a report from each player that describes their valuation function, then (2) use the reports to decide on an allocation, and finally (3) charge each player a monetary payment. The map of step (2) from reports to an allocation is called the *allocation rule* of the mechanism, and the map of step (3) from reports to payments is called the *payment rule*.

Our goal is to design mechanisms that satisfy three properties: (a) a self-interested player who strategically chooses his report in step (1) — in anticipation of the actions of the mechanism in steps (2) and (3) — reports his valuation truthfully; (b) can

be implemented efficiently, i.e., in polynomial time; and (c) when players report truthfully, the resulting allocation is approximately optimal, as measured by the objective function. Requirement (a) is called *incentive compatibility*, and we adopt a particularly strong form of this requirement which we describe in Section 1.3.3. Requirement (c) is parametrized by a desired *approximation ratio* α, and requires that the objective function value for the output allocation be at least a factor α of its maximum over all allocations.[6]

We note that the Vickrey auction of Section 1.2.1 for the single-item allocation problem satisfies all three conditions, with an approximation ratio of 1 for the social welfare objective. In contrast, the instantiation of the Vickrey-Clarke-Groves (VCG) mechanism for combinatorial allocation problem described in Section 1.2.2 satisfies requirements (a) and (c) with an approximation ratio of 1, but does not satisfy (b), implementation in polynomial time, assuming $P \neq NP$. This is because VCG's allocation rule requires computation of the welfare-maximizing allocation of items, which, as explained in Section 1.2.2, is NP-hard even when players report their coverage valuations truthfully.

1.3.3 Incentive Compatibility

In this thesis, we design randomized mechanisms that satisfy a strong form of incentive compatibility, which we refer to as *truthfulness in expectation*. Before describing truthful in expectation mechanisms, we define a player's *utility* from an execution of a mechanism to be his value for the computed allocation (as given by the player's valuation function) less his payment. In the Vickrey auction, for example, the winning player's utility is his value for the item less the second highest bid, and the utility of losing bidders is zero. A randomized mechanism is *truthful in expectation* if each player maximizes his expected utility by bidding his true valuation function, regardless

[6]In some discussions it is notationally convenient to refer to the inverse of this ratio, a number greater than 1. Therefore, as is traditional in the approximation algorithms literature, we refer to both α and $1/\alpha$ as the *approximation ratio*, and make our intention clear in context.

of the bids of others.[7] Randomized mechanisms that satisfy this condition for each flip of their coins, rather than just in expectation, are called *universally truthful*, and deterministic mechanisms that satisfy this condition are simply called *truthful*. The Vickrey auction, as well as the more general VCG mechanism, satisfy this property as discussed in Section 1.2.

1.3.4 Welfare Maximization and VCG

Much of this thesis, with the exception of Chapter 10, is concerned with mechanism design problems where the objective is to maximize social welfare. Welfare maximization problems are central to the study of resource allocation in strategic settings, in part due to the large body of natural problems they represent; we elaborate on their significance in Chapter 3.

Welfare maximization problems are particularly attractive to study from a computational point of view because the economic challenge, as posed in this introduction, has been solved for these problems: The Vickrey-Clarke-Groves (VCG) mechanism, described in the special cases of single-item allocation, combinatorial allocation, and knapsack allocation in Section 1.2, can be phrased generally for an abstract mechanism design problem, and is a deterministic truthful mechanism that maximizes social welfare. This leaves computation as the binding constraint, exposing in the process the conflict between economic and computational considerations.

Unfortunately, the VCG mechanism can not be implemented in polynomial time for many problems, particularly because it requires computation of the optimal allocation. Nevertheless, as we discuss in the rest of this thesis, variants of this mechanism where the set of allocations is "trimmed" to a smaller set, in particular before valuations are solicited from the players, allow the design of many approximation mechanisms for welfare maximization.

[7] Assuming our utility model, truthful-in-expectation mechanisms are better known in the microeconomics literature as *dominant-strategy incentive-compatible*. See Section 2.6 for a discussion.

1.4 Incentive Compatibility and Tractability

As discussed for Combinatorial allocation and Knapsack allocation, many resource allocation problems are known to admit an optimal incentive-compatible mechanism as well as a near-optimal polynomial-time algorithm. However, in most interesting cases the former runs in exponential time, and it is not known how to use the latter as part of an incentive-compatible mechanism. This motivates the central consideration of this thesis: the design of "good" *approximation mechanisms* for resource allocation. These mechanisms should be incentive compatible, should run in polynomial time, and should provide an approximation guarantee near to that of the best (non-incentive-compatible) polynomial-time algorithm for the resource allocation problem at hand. For combinatorial allocation with coverage valuations (Section 1.2.2), for instance, the best we can hope for is an incentive-compatible, polynomial-time mechanism that always outputs an allocation with welfare at least 63% of the maximum possible. For knapsack allocation (Section 1.2.3), this would be an incentive-compatible mechanism that is an FPTAS.

1.4.1 A Naive Approach

Recall from Section 1.3.4 that the Vickrey-Clarke-Groves mechanism is incentive compatible and maximizes welfare, and yet cannot be implemented in polynomial time for problems such as combinatorial allocation and knapsack allocation, among others. This is precisely because VCG insists on the welfare-maximizing allocation, requiring the solution of an NP-hard optimization problem.

This motivates the following approach: Consider "plugging in" a polynomial-time approximation algorithm for welfare maximization — say, one of those mentioned in Section 1.2 — into VCG, in place of the welfare-maximizing allocation rule used in the standard definition of the mechanism. The resulting adaptation of VCG runs in polynomial time, and moreover its approximation guarantee is no worse than that provided by the approximation algorithm. Unfortunately, it was shown by Nisan and Ronen [71] that this naive approach destroys incentive compatibility for all but a very special class of approximation algorithms (discussed in Chapter 3). This failure is a

consequence of the intricate relationship between the payments charged by the VCG mechanism and the players' values for the optimal allocation; this relationship breaks down for most "classical" approximation algorithms in the literature.

1.4.2 Tension

The failure of the naive approach of Section 1.4.1 is emblematic of an underlying tension between incentive compatibility and computational efficiency. Foundational work in mechanism design has shown that incentive compatibility severely restricts the algorithm used for computing the allocation. Most existing techniques for the design of approximation algorithms violate these restrictions. As a result, "good" approximation mechanisms have eluded researchers in algorithmic mechanism design, both for many specific resource allocation problems as well as in general.

We now attempt to provide some intuition to support the apparent incompatibility of many approximation algorithms with incentive compatibility. Approximation algorithms, by definition, are given flexibility to err. Consider the value of a specific player for the allocation output by an algorithm, as a function of that player's report, after fixing the reports of other players. Whereas there is little flexibility in the allocation of an optimal algorithm, fluctuations in the value of the player for the allocation of an approximation algorithm can be almost arbitrary (so long as the approximation guarantee is not violated). For incentive compatibility, a payment must be defined for each possible report of the player, and these payments must compensate for fluctuations in the allocation so that truthful reporting is the player's best strategy. Otherwise, a strategic player can gain by misreporting his private information, effectively exploiting the algorithm's errors. Since these payments make reference only to the player's report, and yet must enforce truthful reporting for an (unknown) private valuation, this is possible only for a restricted class of approximation algorithms, where fluctuations in the allocation satisfy certain conditions.[8]

[8]These algorithms are often called *cycle monotone* [78]. An exactly welfare-maximizing algorithm, as seen in the VCG mechanism, is one example. There are others, some of which are explored in Chapter 3.

Is the tension between incentive compatibility and polynomial-time approximation fundamental, or merely due to limitations of existing techniques for the design of approximation algorithms? While this remains poorly understood in general, a recent series of negative results concerning *deterministic* mechanisms supports the former hypothesis. These results rule out "good" approximation mechanisms that are deterministic for a few resource allocation problems. More common, however, are results that rule out successful application of specific techniques, commonly those based on variations of VCG, for the design of deterministic mechanisms.

1.5 Deterministic vs. Randomized Mechanisms

We now explore the tension between incentive compatibility and computational efficiency, in particular as it relates to the use of randomization. We rationalize the apparent limitations of deterministic mechanisms, and explain how randomization appears to overcome these barriers. This discussion provides the intuition underlying the discoveries of this thesis.

1.5.1 Limits of Deterministic Mechanisms

In order to appreciate the limitations of deterministic mechanisms, we informally examine the constraints placed by incentive compatibility. Incentive compatibility demands that, from each player's perspective, the value of the player for the mechanism's allocation less his payment is maximized by truthful reporting. Roughly speaking, this constrains the mechanism to be "optimal" for each player, simultaneously, with respect to the "range" of his possible misreports. This can be interpreted as a form of local optimality relative to other possible outputs of the mechanism. For example in combinatorial auctions, a player should maximize his value for the package of items received less its price by truthful reporting.

For deterministic mechanisms whose output space is expressive enough to guarantee a good approximation, the constraints placed by incentive compatibility frequently require the solution of an intractable sub-problem: either optimality-in-range for an

individual player is an intractable optimization problem in itself, or, when required simultaneously for all players, implies a relaxed notion of global optimality of the overall allocation that is also impossible to achieve in polynomial time. In both cases, the intractability is due in-part to the "complexity" of the space of discrete structures that describe potential allocations — for example the space of functions mapping items to players in combinatorial allocation, or the space of subsets of projects with total cost at most a budget for knapsack allocation.

1.5.2 The Potential of Randomized Mechanisms

As described in Section 1.5.1, deterministic approximation mechanisms for discrete resource allocation problems are often constrained by incentive compatibility to *exactly* solve a discrete sub-problem of similar complexity as part of their implementation. Allowing the mechanism to *randomize* its choice of allocation adds an additional degree of freedom that can be exploited to relax this requirement. Specifically, we can think of the output of a randomized mechanism as a *lottery* over allocations. While the space of allocations is typically discrete as in the examples of Section 1.2, the space of *lotteries over allocations* is a connected, convex set.

Discrete optimization problems are frequently intractable to solve exactly. Indeed, most NP-hard optimization problems are discrete in nature (see [43] for a selection). In contrast, problems whose solution space is convex are frequently easier to solve (see, e.g. [15]). In fact, approximation algorithms for discrete problems are often based on techniques for solving continuous optimization problems. Perhaps the best example of this is the pervasive application of *linear programming* to the design of approximation algorithms: A discrete, NP-hard optimization problem is first *relaxed* to a linear program that approximately encodes the original problem. The linear program can then be *solved exactly* in polynomial time, and its solution re-interpreted — in an approximate sense — as a solution to the original discrete problem. This approach has led to some of the best known approximation algorithms for many NP-hard problems (see e.g. [84]).

Our use of randomization in algorithmic mechanism design is similar in spirit, as well as at times in its technical content, to the use of continuous techniques in discrete optimization. However, there are important differences. Consider a problem where any incentive-compatible deterministic mechanism with a good approximation guarantee must implicitly solve a discrete and intractable per-player utility-maximization problem as part of its implementation (as described in Section 1.5.1). A randomized mechanism may be designed so that utility maximization over the range of a player's possible misreports is a continuous optimization problem, with a set of lotteries as its feasible region. If the mechanism's allocation algorithm is carefully tuned, this continuous optimization problem may be easier to solve than its discrete counterpart, bypassing an intractability barrier faced by deterministic mechanisms. Unlike a linear-programming relaxation, however, the feasible region of the utility-maximization problem induced by a randomized mechanism must track the output of the mechanism *exactly*, rather than approximately.

1.5.3 VCG-based Mechanisms

This dichotomy between deterministic and randomized mechanisms is particularly intelligible for welfare maximization problems, in particular those where each player's valuation is described by multiple private parameters. (See Section 1.3.1 for an informal definition of multi-parameter problems.) This class includes most resource allocation problems considered in this thesis. For such problems, most incentive-compatible approximation mechanisms that have been succesfully employed are variants of the VCG mechanism.[9] For illustrative purposes, we limit our attention in the remainder of this section to approximation mechanisms based on VCG, and compare and contrast deterministic and randomized variants thereof.

Recall that the incentive-compatible VCG mechanism chooses the allocation that maximizes welfare, and charges each player his externality. This mechanism cannot be implemented efficiently in general, because welfare maximization is NP-hard in many contexts. Consider the following deterministic modification of VCG: the set of

[9]Moreover there is theoretical evidence that, for many of these problems, mechanisms based on VCG are the only truthful mechanisms. We elaborate on this in Chapter 3.

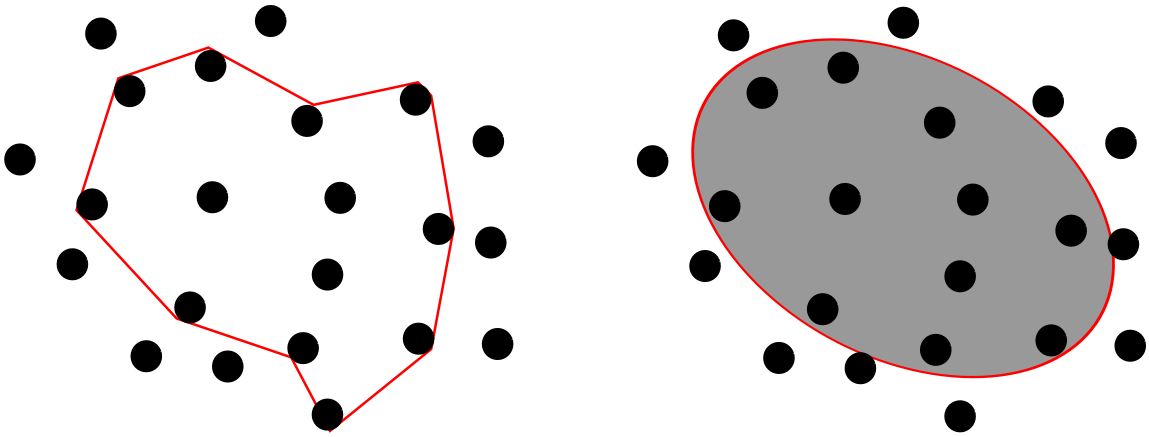
allocations is "trimmed" to a smaller set in some way, in particular before valuations are solicited from the players, and then the VCG mechanism is applied to the smaller problem of maximizing welfare over the "trimmed" set of allocations. If the trimmed set, which we refer to as the *range* of the mechanism, is "small" enough to allow polynomial-time optimization, and at the same time is "big" enough to adequately approximate the original set of allocations,[10] the result is a polynomial-time and incentive-compatible approximation mechanism.

An example of this approach was used by Dobzinski et al. [32] for combinatorial allocation. Specifically, the range of their mechanism consists of those allocations that assign each player at most a single item, as well as those that assign all items to a single player. Finding the best allocation in this range boils down to finding the best matching of items to players, which is possible in polynomial time. Therefore, the instantiation of VCG for this range can be implemented in polynomial time, and is incentive compatible. Moreover, when valuations are suitably restricted this mechanism guarantees a non-trivial, though super-constant, fraction of the maximum social welfare.

Their limited successes notwithstanding, these deterministic VCG-based approximation mechanisms face the same intractability barriers described in Section 1.5.1. In this context, an alternative interpretation of the intractability barriers is instructive: Any "trimmed" range of allocations that "big" enough to adequately approximate the original (untrimmed) set of allocations is also big enough to "inherit" the intractability of the original (untrimmed) discrete optimization problem. Statements of this form have been proven for variants of combinatorial allocation [29, 17, 26], among other problems.

We bypass these difficulties by designing the range of a mechanism as a family of *lotteries over allocations*, rather than (deterministic) allocations. As an example, we exploit in Chapter 5 the following range of lotteries for the combinatorial allocation problem: A lottery is in our range if it associates with each item a distribution over

[10] In particular, because the trimmed set does not depend on player valuations, it should contain an "approximately optimal" allocation for each possible valuation profile.



(a) Range of a deterministic approximation mechanism.

(b) Range of a randomized approximation mechanism.

Figure 1.2: Deterministic vs Randomized VCG-based mechanisms. Each black point is an encoding of an allocation in Euclidean space. In (a), the range is the circumscribed set of allocations. In (b), the range, a family of lotteries over allocations, is depicted as the gray region inside the convex hull of allocations.

players, with no player's probability exceeding 63%, and assigns the items independently according to their respective distributions. For such a range of lotteries fixed independently of player valuations, the following randomized analogue of the VCG mechanism is incentive compatible: Solicit player valuations, then find the lottery in the range that maximizes expected social welfare, and finally output a sample from the winning lottery and charge each player his expected externality. These mechanisms are termed *maximal in distributional range*, and we explore them formally in Chapter 3.

The results of this thesis demonstrate that these randomized adaptations of VCG are more "powerful" than their deterministic counterparts; specifically, they enable better approximation guarantees in polynomial time. Their power stems from their more-permissive design space. In settings where a range of allocations that is "big enough" for a good approximation is necessarily too combinatorially "complex" for exact optimization, it may be possible to design a family of lotteries that is "big enough" and yet "simple enough". We depict this dichotomy in Figure 1.2.

1.6 Contributions of this Thesis

Conceptually, the contributions of this thesis are twofold:

1. We design randomized mechanisms for a variety of fundamental resource allocation problems that are incentive compatible, run in polynomial time, and match the approximation guarantee of the best (non-incentive-compatible) polynomial-time algorithm. In most cases, our mechanisms improve on the best previously-known approximation guarantee of an incentive-compatible and polynomial-time mechanism by a constant or super-constant factor.

2. We develop new techniques that utilize randomization to combine incentive compatibility and computational tractability. In the process, we advocate the hypothesis that randomization can reconcile incentive compatibility and computational tractability in settings where deterministic mechanisms are apparently — and in some cases provably — limited.

We summarize a selection of our results for specific resource allocation problems in Table 1.1. All our mechanisms are truthful in expectation and run in time polynomial in natural parameters of their input.[11] For each problem, we list the approximation guarantee of our mechanism, the approximation guarantee of the best polynomial-time and truthful-in-expectation mechanism from previous work, and the approximation guarantee of the best (non-incentive-compatible) polynomial-time algorithm. However, we believe that our contributions are best viewed through our techniques. Therefore, we group together results of a similar technical flavor, and describe each suite of techniques before overviewing the specific results it obtains.

1.6.1 The Convex Rounding Framework

Algorithmic mechanism design is difficult because incentive compatibility severely limits how the algorithm can compute an outcome, which prohibits use of most of the

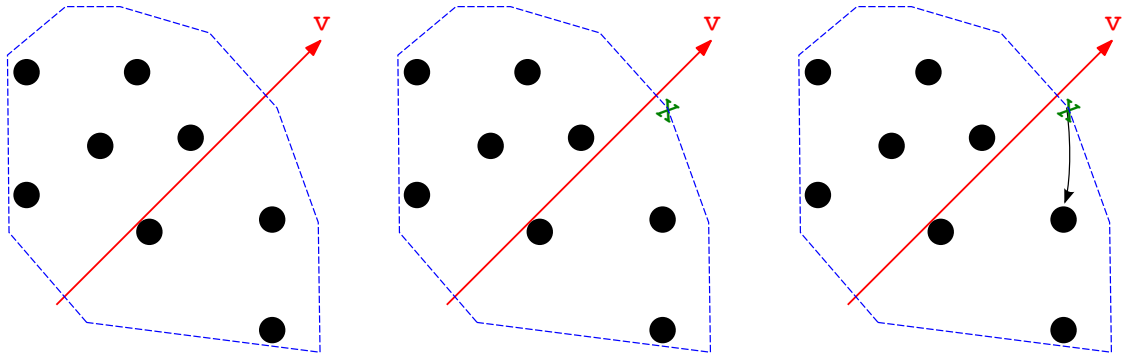
[11] In many cases, the natural parameter is the length of the description of the input. For some of the problems we consider, however, parameters such as the number of players and/or resources are more natural.

Problem	Prior Work (truthful)	Our Work (truthful)	Best Possible (non-truthful)
Combinatorial Auctions (MRS valuations)	$\left(O\left(\frac{\log m}{\log\log m}\right)\right)^{-1}$	0.63	0.63
Combinatorial Public Projects (MRS valuations)	$(O(\sqrt{m}))^{-1}$	0.63	0.63
Multi-unit Auctions	0.5	$1-\epsilon$	$1-\epsilon$
Knapsack Allocation	0.5	$1-\epsilon$	$1-\epsilon$
Scheduling with Deadlines	—		
...*	—		
Minimum Makespan Scheduling	2	$1+\epsilon$	$1+\epsilon$

Table 1.1: Table of results.
Each table entry is an approximation ratio. For combinatorial auctions and public projects, m denotes the number of items and projects, respectively. The variable ϵ denotes an arbitrary positive constant. The wildcard * denotes any binary packing problem of polynomial dimension that admits an FPTAS, as defined in Chapter 8.

ingenious techniques developed for the design of "classical" approximation algorithms. This phenomenon, described in Section 1.4.2, motivates the following question: can these "classical" techniques be adapted to the design of incentive-compatible mechanisms?

In Chapter 4, we consider what is possibly the dominant "classical" approach for the design of (non-truthful) approximation algorithms for discrete and NP-hard optimization problems. This approach, depicted in Figure 1.3, proceeds as follows: The discrete problem is first *relaxed* to a continuous optimization problem — typically a linear program — that approximately encodes the original problem. This *fractional relaxation* is then *solved* in polynomial time; for a welfare maximization problem, this means that social welfare — or, rather, a "fractional extension" of the social welfare function — is maximized over points of the relaxation. Finally, the resulting fractional solution is *rounded*, via a *rounding algorithm*, to a "nearby" solution that is feasible for the original discrete problem. With rare exceptions, such algorithms can not be used as part of a truthful mechanism, mainly due to a lack of structure in their rounding algorithm that leaves them open to strategic manipulation. Specifically, the outcome from *solving* the relaxation and then *rounding* the fractional solution may "fluctuate"



(a) Relaxation (b) Solving (c) Rounding

Figure 1.3: The relax-solve-round approach using a linear program. Each black point is an encoding of an allocation in Euclidean space. The polygon depicts the boundary of a linear programming relaxation. The arrow labeled v depicts a linear objective function.

almost arbitrarily as a player changes his report; as described in Section 1.4.2, this creates opportunities for strategic manipulation.

We overcome this difficulty by employing a simple yet powerful idea: we optimize welfare *directly over the (random) output of the rounding algorithm*, rather than over the *input* to the rounding algorithm. Observe that a (randomized) rounding algorithm maps each point in the fractional relaxation to a lottery over allocations, and therefore the rounding algorithm's *range* of outputs is a family of lotteries; this is depicted in Figure 1.4. Our framework proposes the allocation rule that computes the "best" lottery in the rounding algorithm's range; specifically, the lottery that maximizes expected social welfare. As discussed in Section 1.5.3, an allocation rule of this form is *maximal in distributional range*, and can be generically supplemented with payments to yield a truthful-in-expectation mechanism.

A difficulty remains, however: Whereas a fractional relaxation is designed so that it can be solved efficiently, finding the best possible output of a rounding algorithm is typically NP-hard. To address this, we propose a family of rounding algorithms with additional structure. A *convex rounding algorithm* is one with the following remarkable property: the expected social welfare from rounding a fractional solution x is a concave function of the variables of x. Finding the welfare-maximizing lottery

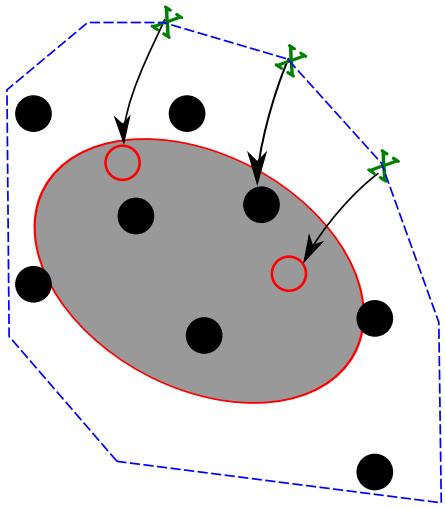


Figure 1.4: The range of a randomized rounding algorithm.
Rounding maps each point in the linear programming relaxation to a lottery over allocations. The resulting family of lotteries is depicted as the gray region in the convex hull of allocations.

in a convex rounding algorithm's range is a convex optimization problem, which in most cases can be solved efficiently.

This framework can be interpreted as an extension of the dominant approach for approximation algorithm design to our more restrictive computational model. An algorithm designer seeking to build a truthful mechanism now faces a familiar, albeit more particular, task: the design of a rounding algorithm satisfying an additional condition.

Combinatorial Auctions

As our first application of the convex rounding framework, we consider the combinatorial allocation problem overviewed in Section 1.2. In a mechanism design context, this problem is referred to as *combinatorial auctions*. The design of efficient, incentive-compatible combinatorial auctions is a central research challenge in algorithmic mechanism design. This is due to the broad applicability of the problem, as well as due to its fundamental nature from a theoretical perspective. Indeed, much of

the progress in the design of approximation mechanisms has been motivated by the study of combinatorial auctions, and techniques first developed for this problem have since been exported elsewhere.

There are no useful approximation algorithms for the most general variant of combinatorial auctions; that where a player's value as a function of the bundle of items is an arbitrary set function. In reality, however, valuation functions in combinatorial auctions are frequently structured. As an example, consider the coverage valuations presented in Section 1.2.2 and depicted in Figure 1.1.

In Chapter 5, we consider a well-motivated variant of combinatorial auctions: player valuations are restricted to a generalization of the coverage valuations of Section 1.2.2. We design a convex rounding algorithm for this problem, leading to a truthful-in-expectation mechanism that runs in polynomial time, and guarantees a $1 - 1/e \approx 63\%$ approximation to the optimal social welfare. This approximation factor matches the best possible, even by a (non-truthful) approximation algorithm. The result of Chapter 5 is the first constant-factor approximation for a variant of combinatorial auctions where welfare maximization is NP-hard.

Combinatorial Public Projects

Our second application of convex rounding is to the *combinatorial public projects* problem. In this problem, a public planner must choose from a set of projects to undertake, subject to a constraint on the number of projects chosen, and a set of self-interested players have private valuations over packages of projects. Combinatorial public projects has emerged as the paradigmatic "hard problem" of algorithmic mechanism design, due to strong impossibility results concerning deterministic mechanisms for variants of this problem.

As in combinatorial auctions, this probem only admits non-trivial approximation algorithms if considered with restricted valuations. In Chapter 6, we consider combinatorial public projects where player valuations must lie in a certain class generalizing coverage valuations. We design a convex rounding algorithm for this problem, leading to a truthful-in-expectation mechanism that runs in polynomial time, and guarantees a $1 - 1/e \approx 63\%$ approximation to the optimal social welfare. This approximation

factor matches the best possible, even by a (non-truthful) approximation algorithm. The result of Chapter 6 is the first truthful-in-expectation and polynomial-time mechanism to achieve a constant-factor approximation for a natural NP-hard variant of combinatorial public projects, and stands in contrast to the body of existing negative results for this problem.

1.6.2 The Linear Perturbation Framework

Our next technique is motivated by intuition from the celebrated field of *smoothed analysis of algorithms*. Smoothed analysis was developed to explain the efficiency in practice of algorithms that run in exponential time on a worst-case input. Models of smoothed analysis mildly relax the worst-case paradigm by assuming that inputs are "imprecise"; specifically, worst-case inputs are subjected by nature to a small amount of "random noise" before being presented to the algorithm. Provided the noise is "sufficiently random", many tasks that are intractable in the worst case can be solved efficiently (in expectation) for the "noisy" input. Such problems are said to have *polynomial smoothed complexity*. The knapsack allocation problem of Section 1.2.3 is one example: If suitable "random noise" is added to each player's value for each project, the resulting "noisy" social welfare function can be maximized in expected polynomial time. We refer to the map from the original input to the "noisy" input as a *perturbation*.

With the realization that some intractable resource allocation problems become tractable when randomly perturbed, we turn the smoothed analysis paradigm on its head. Rather than postulating a "natural" perturbation that explains tractability observed in practice, we ask whether perturbations can be *designed* to yield both tractability and incentive compatibility. Specifically, consider a *perturbation-based allocation algorithm* that first perturbs the valuation functions of players in some way, and finds an allocation that maximizes the resulting perturbed social welfare function *exactly*. For problems with polynomial smoothed complexity, the perturbation can be designed so that this algorithm can be implemented efficiently (in expectation), and yet the perturbed welfare function is "close enough" to the original welfare function

for the algorithm to guarantee a good approximation (in fact, an FPTAS). Naievely, the perturbation-based allocation algorithm also appears amenable to incentive compatibility — it differs from the allocation rule of the VCG mechanism only in its pre-application of the perturbation.

On further examination, a difficulty remains: perturbing a player's valuation does not, in general, preserve incentive compatibility. Indeed, a perturbed misreport may better represent the player's true preferences than the perturbed version of the player's true valuation function. We overcome this difficulty by introducing a *duality* between perturbations of the social welfare function and perturbations of the allocations. We observe that, when perturbations of the social welfare function are *linear*, the perturbed welfare of an allocation can be re-interpreted as the un-perturbed welfare of a *perturbed allocation*. When certain technical conditions are satisfied, the perturbed version of an allocation can be interpreted as a lottery that mixes the original allocation with some other allocations with small probability. Adopting this "dual" perspective, the perturbation-based allocation algorithm maximizes expected social welfare over the set of lotteries corresponding to the perturbed allocations. This is a maximal-in-distributional-range allocation algorithm (Section 1.5.3), and can therefore be supplemented with VCG payments to yield a truthful-in-expectation mechanism.

A Black Box Result

Armed with the linear perturbation framework, we turn our attention to those resource allocation problems with polynomial smoothed complexity. For which of these problems can we combine polynomial smoothed complexity and incentive compatibility? In particular, can we define a large subclass of these problems so that conditions required for application of the linear perturbation framework are satisfied, resulting in efficient and incentive-compatible mechanisms that attain the best possible approximation ratio?

We answer this question in the affirmative for a large and natural class of resource allocation problems, defined abstractly. We consider welfare maximization problems that can be encoded as *binary packing problems with polynomial-dimension*, defined

formally in Chapter 8. In an instance of such a problem, each allocation can be encoded via a polynomial number (in the length of the representation of the instance) of binary decisions, and each player's valuation function is linear in the variables of this encoding. Moreover, the set of allocations is "downwards closed", in that a positive decision in an allocation can be reversed to form another feasible allocation. Examples of these problems abound in resource allocation, one of which is the knapsack allocation problem discussed in Section 1.2.3: each project's inclusion in a solution corresponds to a 0/1 decision, and a project may be removed from a solution without affecting feasibility.

Using the linear perturbation framework of Chapter 7, we show in Chapter 8 that a fully polynomial time approximation scheme (FPTAS)[12] for any problem in this class can be converted, generically and in polynomial time, to a truthful-in-expectation mechanism that is an FPTAS. This is a "black-box" reduction from incentive-compatible mechanism design to approximation algorithm design, in that it assumes nothing about the approximation algorithm beyond its approximation guarantee. This is the first such reduction for a non-trivial class of multi-parameter mechanism design problems, and suggests an intriguing possibility of more general black-box reductions in algorithmic mechanism design.

The main result of Chapter 8 is possible because prior work in smoothed analysis [9, 79] has already established that problems in our class that admit an FPTAS also have polynomial smoothed complexity, relative to some assumptions about the smoothing perturbation. However, application of the linear perturbation framework to obtain an incentive-compatible mechanism requires our perturbation to be linear, and moreover to "dualize" to a perturbation mapping each allocation to a lottery over allocations. As the main technical contribution of this chapter, we exploit the structure of binary packing problems to show that these conditions can be satisfied simultaneously, enabling our black-box reduction.

[12] A *fully polynomial time approximation scheme (FPTAS)* for a maximization problem takes as input an instance and an approximation parameter ϵ, and returns a feasible solution with objective function value at least $1 - \epsilon$ times that of an optimal solution, in time polynomial in the size of the instance and in $1/\epsilon$.

Multi-unit Auctions

In Chapter 9, we consider welfare maximization in *multi-unit auctions*. In this problem, a large (exponential) number of identical items must be allocated among a set of competing players, and each player is equipped with a non-decreasing valuation function that maps the number of items he receives to his value.[13] This problem captures many scenarios where a homogenous resource must be divided among multiple players, such as bandwidth, machine processing time, or electrical power. Despite intense study prior to our work, and despite admitting a (non-truthful) FPTAS, multi-unit auctions have resisted approximation mechanisms obtaining better than a 50% approximation guarantee.

Welfare maximization in multi-unit auctions admits an encoding as a binary packing problem, and yet escapes the black-box result of Chapter 8. This is because its formulation as a binary packing problem has an exponential number of variables, due to the exponential number of items being allocated. The high dimensionality of the problem poses two difficulties in applying our linear perturbation framework: the results of smoothed analysis no longer imply polynomial smoothed complexity, and the perturbations we design in Chapter 8 can not be directly applied to an instance of this problem efficiently.

The main result of Chapter 9 overcomes these difficulties to design a truthful-in-expectation FPTAS for multi-unit auctions. We again use the linear perturbation framework of Chapter 7: we design linear perturbations that exploit the structure of multi-unit auctions to recover polynomial smoothed complexity, and at the same time can be applied efficiently to solutions of exponential dimension.

1.6.3 Techniques for Single-Parameter Problems

In Chapter 10, we turn our attention to an apparently simpler class of problems, where each player's valuation is naturally described via a single real number. These problems, known as *single-parameter problems* and defined formally in Chapter 10,

[13] We assume no particular representation of this function, and only require that it can be queried efficiently at any number of items.

have received dedicated study in algorithmic mechanism design. This is because they occur naturally in a variety of contexts (the single-item auction is one example), and moreover admit more positive results for more objective functions than their multi-parameter counterparts.

We consider single-parameter problems in a machine-scheduling context. Such a problem features n *jobs*, each with a publicly known *size*, and m *machines* (the players). An allocation is a *schedule* that assigns each job to one of the machines. Each machine holds privately its *speed*, measured in job size per unit time. A mechanism for a scheduling problem may reward machines with payments to incentivize truthful revelation of their speeds. Adopting the adage "time is money", a machine's utility is naturally defined as the payment it receives from a mechanism less the time it spends processing its assigned jobs.

Many objectives are possible in a scheduling problem. In our primary application, we consider a mechanism designer interested in minimizing the *makespan* of the schedule, which is defined as the maximum, taken over all machines, of the time taken by the machine to process all jobs assigned to it. Equivalently, the makespan of a schedule is the time at which all jobs have been completed.

Chapter 10 develops and applies several flexible techniques for the design of incentive-compatible mechanisms for single-parameter scheduling problems. Next, we overview these techniques and then describe our results.

Techniques

Our primary design framework in this thesis, that of maximal in distributional range algorithms, does not appear to be of much direct use for problems with a non-welfare objective such as makespan. Nevertheless, we observe that the same basic approach, that of exact optimization over a set of lotteries, can be adapted for non-welfare objectives in some single-parameter contexts, with additional work. For the makespan problem, for example, we show that an algorithm that fixes a set of lotteries over schedules independently of reported speeds, and computes the lottery that minimizes expected makespan, is truthful in expectation *provided that ties between lotteries are broken in a "truthfulness-preserving" manner.* Satisfying the additional tie-breaking

requirement in polynomial time is non-trivial, but we present a generic way to accomplish it efficiently for scheduling problems. We emphasize that the viability of this approach, in particular for non-welfare objectives, appears limited to single-parameter problems due to their more permissive space of incentive-compatible mechanisms.

The design of a family of lotteries over which makespan, as well as the other objectives we consider, can be exactly optimized raises additional technical challenges. We introduce a number of techniques to construct a range for our mechanism that is both big enough to guarantee an approximately optimal schedule, and yet simple enough to enable polynomial-time optimization. We defer discussion of most of the techniques to Chapter 10, but we highlight one of them here that is representative. We group jobs of a similar size together, and *randomly shuffle* the jobs in each group. In essence, shuffling replaces each job in a group with the "average" of that group, in expectation. A schedule of the "shuffled jobs" is a simpler object, with many jobs essentially identical, and moreover corresponds to a lottery over schedules of the original, unshuffled jobs. When combined with additional techniques, shuffling the jobs yields a family of lotteries over which exact optimization of makespan, as well as some other objectives, is possible in polynomial time.

Results

The main result of Chapter 10 is for the makespan problem. Despite intense study prior to our work, and despite admitting a (non-truthful) Polynomial Time Approximation Scheme (PTAS),[14] the best polynomial-time and truthful-in-expectation mechanism for the makespan problem was a 2-approximation. We use the techniques outlined in the preceding discussion to develop a truthful-in-expectation PTAS for the makespan problem. Moreover, we apply the same techniques to other objectives, namely minimizing a norm of the machine completion times and maximizing the minimum completion time. In both cases we obtain a truthful-in-expectation PTAS,

[14] A PTAS for a minimization problem is an algorithm that takes as input an instance of the problem and a parameter $\epsilon > 0$, runs in time polynomial in the representation of the instance, and returns a solution with objective at most $1 + \epsilon$ of the minimum possible.

whereas no truthful-in-expectation and polynomial-time mechanism with even a constant approximation ratio was previously known for either problem.

1.7 Related Work

We now place the contributions of this thesis in the proper context. We begin in Section 1.7.1 with the basics of mechanism design, as they relate to this thesis. Section 1.7.2 discusses work that characterizes the structure of incentive-compatible mechanisms. Finally, Section 1.7.3 discusses both positive and negative results that directly pertain to the design of computationally-efficient and incentive-compatible approximation mechanisms. Throughout, we restrict our attention to the most relevant references, and defer some details to relevant chapters. We also omit foundational work in approximation algorithms, as its discussion is beyond the scope of this thesis, and instead refer the reader to a text on the subject such as [84].

1.7.1 Mechanism Design Basics

Our work relies on the rich theory of mechanism design, most of which evolved during the 1960s through the 1980s. We mention here only the most pertinent work in mechanism design to this thesis. A survey of mechanism design concepts intended for computer scientists is in [70]. For more detail, we refer the reader to resources on microeconomics (e.g. [63]) and auction theory (e.g. [57, 65, 55]).

The study of mechanisms that use monetary payments was initiated by Vickrey [85]. Ths single-item Vickrey auction was then generalized by Clarke [21] and Groves [46] to more general settings, culminating in the Vickrey-Clarke-Groves (VCG) mechanism for welfare maximization that forms the starting point for much of the work in this thesis.

As we discuss in Section 2.6, most work on mechanism design in the microeconomics community considers the Bayesian setting, where player valuations are assumed to be drawn from a known distribution, and incentive compatibility is only

required to hold in expectation with respect to this distribution. Bayesian incentive-compatible mechanisms are more permissive than the class of dominant-strategy incentive-compatible mechanisms considered in this thesis. Nevertheless, some results in the Bayesian setting have analogues in our setting; a notable example is Myerson's characterization of Bayesian incentive-compatible mechanisms for single-item allocation [68]. For more on mechanism design in the Bayesian setting, we refer the reader to [63].

1.7.2 Characterizations of Incentive Compatibility

Work in both the mechanism design and algorithmic mechanism design communities has sought to *characterize* the space of incentive-compatible mechanisms in various contexts. Myerson [68] characterizes Bayesian incentive-compatible mechanisms for the single-item auction problem: such a mechanism must have a *monotone* allocation rule, which means that the probability a player wins the item is non-decreasing in his valuation, and its payment rule is uniquely determined up to "pivot terms". Myerson's monotonicity characterization extends to dominant-strategy incentive-compatible mechanisms for the class of single-parameter problems, and this was made explicit by Archer and Tardos [3].[15] We exploit this characterization when we consider mechanisms for single-parameter scheduling problems in Chapter 10.

Various characterizations of incentive-compatible mechanisms for multi-parameter mechanism design problems are known. Whereas some of these characterizations have been used to prove impossibility results in algorithmic mechanism design, they will be of little technical use in this thesis. Nevertheless, we now briefly overview the most notable such characterization results for perspective. Absent any assumption on how, if at all, valuations are restricted, Rochet [78] defines a property of an allocation rule, *cycle-monotonicity*, and shows that an allocation rule can be supplemented with payments to yield a truthful mechanism if and only if it is cycle-monotone. A stronger, necessary condition called *weak-monotonicity* was posed by Bikhchandani et al. [11],

[15] We also note that the monotonicity characterization is implicit in the earlier work of Mirrlees [66] and Spence [82]. We thank Aaron Archer for pointing this out.

who prove its sufficiency in some domains. Saks and Yu [81] then showed that weak-monotonicity is equivalent to cycle-monotonicity in domains where players' valuation spaces are convex, as is the case for most natural problems in mechanism design. The most relevant characterization of all for this thesis, however, is due to Roberts [77]: for the general multi-parameter mechanism design problem, where a player's valuation is allowed to be an arbitrary function from outcomes to real numbers, Roberts shows that deterministic truthful mechanisms are precisely the set of *affine maximizers*, close relatives of the VCG mechanism that are discussed in Chapter 3.

1.7.3 Computationally-Efficient Mechanisms

The existence of a conflict between incentive compatibility and computational efficiency was first suggested by the seminal work of Nisan and Ronen [72], who coined the term *algorithmic mechanism design*. Since then, much work in the theoretical computer science community has focused on establishing both upper and lower bounds on the approximation ratio of the best incentive-compatible and polynomial-time mechanism for various resource allocation problems.

The main consideration of this thesis is the study of the *conflict* between incentive compatibility and computational efficiency, rather than the challenges in achieving either alone. Therefore, we only mention work concerning problems for which the approximation ratio of the best known incentive-compatible and computationally-efficient mechanism is (or was prior to the work) worse than the approximation ratios of *both* the best incentive-compatible mechanism and the best polynomial-time algorithm.[16]

Next, we present positive and negative results from related work in our model, and close with a description of some recent positive results in the more-permissive Bayesian setting.

[16]As intimated in the rest of this introduction, for all problems we consider, and for all welfare maximization problems generally, the best incentive-compatible mechanism has an approximation ratio of 1. Therefore, the "target" in algorithmic mechanism design is typically the approximation ratio of the best polynomial-time algorithm.

Positive Results and Techniques

The first non-trivial approximation mechanisms that achieve an approximation guarantee near or equal to that of the best (non-truthful) approximation algorithm were discovered for single-parameter mechanism design problems and slight generalizations thereof. We mention three examples: The deterministic truthful mechanism of Lehmann et al. [61] for combinatorial auctions when players are *single-minded*[17] matches the approximation guarantee of the best (non-truthful) approximation algorithm for that problem; Archer and Tardos [3] design a randomized truthful-in-expectation mechanism for a single-parameter scheduling problem — the main problem considered in Chapter 10 — whose approximation guarantee is a constant factor worse than the best possible (non-truthfully) in polynomial time; and, in a remarkably general result, Briest et al. [16] show that a fully polynomial time approximation scheme (FPTAS) for a single-parameter welfare-maximization problem can be converted to a truthful mechanism that is an FPTAS.[18]

Positive results for multi-parameter problems, and in particular results that approach the best approximation guarantee possible in polynomial time, have been more elusive, and particularly so when the focus was on deterministic mechanisms. A central idea, adapting VCG by "trimming" its range of allocations as described in Section 1.5.3, was posed by Nisan and Ronen [71]. Deterministic truthful mechanisms based on this idea, specifically the *maximal-in-range* mechanisms described in Chapter 3, subsequently found application: to combinatorial auctions with unrestricted valuations by Holzman et al. [51], to combinatorial auctions with *complement-free*[19] valuations by Dobzinski et al. [32], and to multi-unit auctions by Dobzinski and Nisan [30]. The approximation guarantee of both results [51, 32] for combinatorial auctions is a super-constant factor away from the best approximation ratio possible

[17] In combinatorial allocation, a player is single minded if his valuation function is non-zero for only a single bundle of items. Note that this a slight generalization of single-parameter problems because the bundle for which the player's valuation function is non-zero is also private.

[18] We note that the main result of Chapter 8 generalizes the result of Briest et al. [16] to multi-parameter problems, under mild additional conditions.

[19] In combinatorial allocation, we say a player has complement-free, or *sub-additive*, valuations if his value for the union of two bundles of items is no more than the sum of his values for the two original bundles.

(non-truthfully) in polynomial time for the respective setting. The result of [30] for multi-unit auctions is a constant-factor approximation mechanism, whereas a non-truthful FPTAS exists.

Progress in the design of approximation mechanisms for multi-parameter problems accelerated with the emerging use of randomization. The starting point is the breakthrough result of Lavi and Swamy [60]. They show that if a welfare maximization problem can be written as a linear program satisfying certain conditions, a truthful-in-expectation and polynomial-time mechanism with an approximation ratio equal to the linear program's *integrality gap* can be generically derived. They subsequently apply this idea to derive a truthful-in-expectation mechanism for combinatorial auctions with unrestricted valuations that matches the best approximation guarantee possible in polynomial time for that problem, and the first truthful-in-expectation and polynomial-time constant-factor approximation mechanism for multi-unit auctions, among other results.

Following the work of Lavi and Swamy, Dobzinski et al. [33] strengthen their result for combinatorial auctions with unrestricted valuations by designing a polynomial-time randomized mechanism that is universally truthful (recall Section 1.3.3), and also matches the best approximation guarantee possible in polynomial time. Universally-truthful randomized mechanisms with a polylogarithmic approximation ratio were also described for variants of combinatorial auctions with restricted valuations by Dobzinski et al. [33] and Dobzinski [25], though in both cases the best approximation ratio possible in polynomial time for the considered problem is a constant.

As evidenced by the preceding discussion, polynomial-time incentive-compatible mechanisms that approach the approximation guarantee of the best non-truthful algorithms have eluded researchers. With the exception of combinatorial auctions with unrestricted valuations, no such "optimal" mechanisms for non-trivial multi-parameter mechanism design problems were known prior to the work presented in this thesis.

A note is in order on this history of maximal-in-distributional-range mechanisms, the primary protagonists in this thesis described in Chapter 3. These mechanisms were formally proposed for the design of approximation mechanisms, and given their current name, by Dobzinski and Dughmi [27]. However, they have appeared implicitly

in prior work. For example, the mechanisms designed by Lavi and Swamy [60] are, in retrospect, maximal in distributional range.

Negative Results

As the limitations of existing techniques for the design of approximation mechanisms became apparent, conditional impossibility results that bound the approximation guarantee of polynomial-time mechanisms began to emerge. While these impossibility results explained the lack of progress in the design of approximation mechanisms for many problems, at first most applied only to deterministic mechanisms based on VCG — specifically the maximal-in-range mechanisms discussed in Chapter 3. Dobzinski and Nisan [29] show a super-constant lower bound on the approximation ratio of deterministic, VCG-based, polynomial-time mechanisms for combinatorial auctions with *submodular*[20] valuations, and Buchfuhrer et al. [17] strengthen the lower bound and generalize it to some other classes of restricted valuations. Dobzinski and Nisan [30] show that deterministic and polynomial-time VCG-based mechanisms for multi-unit auctions cannot guarantee better than 50% of the optimal social welfare, despite the existence of a non-truthful FPTAS.

The first result to *separate* the best approximation guarantee of a deterministic, truthful, and polynomial-time mechanism from that of the best (non-truthful) approximation algorithm is by Lavi et al. [59]; they show that, for a modification of multi-unit auctions, no deterministic, truthful, and polynomial-time mechanism has an approximation ratio better then 50%, despite the existence of a (non-truthful) FPTAS for that problem. More recently, quantitatively stronger separation results for natural problems emerged. Papadimitriou et al. [74] define the combinatorial public projects problem (recall Section 1.6.1), and for a variant of the problem with submodular valuations show that no deterministic, polynomial-time, and truthful mechanism has a constant approximation ratio, despite the existence of a (non-truthful) 63%-approximation algorithm. Dobzinski [26] proves similar impossibility results for combinatorial auctions and a slightly different variant of combinatorial public projects,

[20] A player's valuation function for bundles of items is *submodular* if it satisfies "diminishing marginal returns", in a technical sense described in Chapter 5.

both with submodular valuations, ruling out constant-factor approximation by a deterministic and truthful mechanism in polynomial time despite the existence of a constant-factor approximation algorithm for both problems.

The preceding collection of impossibility results for deterministic mechanisms motivates our focus on randomization. Until very recently, there were no known impossibility results for polynomial-time truthful-in-expectation mechanisms, and general positive results seemed plausible. This changed with the work of Dobzinski [26], who showed a super-constant lower bound on the approximation ratio of polynomial-time and truthful-in-expectation mechanisms for a variant of combinatorial public projects for which a (non-truthful) polynomial-time constant-factor approximation algorithm exists. Subsequently, Dughmi and Vondrák [38] proved similar separation results for variants of combinatorial auctions and combinatorial public projects, separating the approximation guarantee of truthful-in-expectation and polynomial-time mechanisms by a super-constant factor from the best possible in polynomial time for those problems.

Other Models

The task of reconciling incentive compatibility and computational efficiency appears easier in the more-permissive Bayesian setting (described in Section 2.6), where player valuations are drawn from a prior distribution known to the players and to the mechanism. Hartline and Lucier [48] show that for every single-parameter welfare maximization problem in a Bayesian setting, an arbitrary approximation algorithm can be made Bayesian incentive compatible in polynomial time without degrading the expected approximation factor. Under assumptions on the Bayesian prior, this result was recently extended to multi-parameter problems by Bei and Huang [8] and Hartline et al. [47].

1.8 Tips for Reading this Thesis

1.8.1 Prerequisites

The main prerequisite for this thesis is a facility with the basic techniques for the design of approximation algorithms, at the level of Vazirani [84]. A familiarity with the basics of mechanism design and auction theory is very helpful, though we present a self-contained treatment of the relevant preliminaries in this thesis. A survey of mechanism design concepts by Nisan [70], intended for computer scientists, is particularly suited for appreciation of the results in this thesis. Additional detail can be found in a standard reference on microeconomics such as Mas-Colell et al. [63].

We also assume some familiarity with the basics of the following topics: The theory of NP-completeness, for which Garey and Johnson [43] and Arora and Barak [4] are standard references; the theory of convex optimization at the level of Boyd and Vandenberghe [15]; and combinatorial optimization and matroid theory, for which our preferred reference is Korte and Vygen [56].

1.8.2 Outline

Chapters 2 and 3 present the technical and conceptual background underlying our results. Specifically, Chapter 2 defines basic concepts from mechanism design and approximation algorithms, and formalizes the computational and economic requirements we place on approximation mechanisms designed in this thesis. Chapter 3 motivates the detailed study of welfare maximization problems, presents related background and preliminaries, and advocates the class of maximal-in-distributional-range mechanisms as a design framework for these problems.

The remainder of this thesis splits our contributions in three parts, grouped by common techniques. Part II begins by proposing the convex rounding framework in Chapter 4, and this framework is applied to combinatorial auctions and combinatorial public projects in Chapters 5 and 6, respectively. Part III begins by proposing the linear perturbation framework in Chapter 7, and this framework is applied to an

abstractly-defined class of mechanism design problems in Chapter 8, and to multi-unit auctions in Chapter 9. Finally, Part IV proposes new techniques and mechanisms for single-parameter scheduling problems in Chapter 10.

1.8.3 Dependencies

Chapter 2 is a prerequisite for all later chapters. Chapter 3 is a prerequisite for all later chapters except Chapter 10. Chapter 4 is a prerequisite for both Chapters 5 and 6. Finally, the chapters of Part III should be read in order.

1.9 Bibliographic Notes

Most results presented in this thesis appear in one of the following previously published works: [37], [34], [35], [27], and [24]. The convex rounding framework (Chapter 4) and its application to combinatorial auctions (Chapter 5) appear in [37]. The application of convex rounding to combinatorial public projects (Chapter 6) appears in [34]. The linear perturbation framework (Chapter 7) and the resulting black-box result for binary packing problems (Chapter 8) appear in [35]. The result for multi-unit auctions (Chapter 9) appears in [27], though with a substantially different proof than that presented in this thesis. The results for single-parameter scheduling problems (Chapter 10) appear in [24].

We note that the formal definition of maximal in distributional range approximation mechanisms (Chapter 3) appears in [24], though these mechanisms have implicitly been used before as noted in our discussion of related work. The properties of maximal in distributional range algorithms proved in Chapter 3 appear in [27, 35, 37].

Chapter 2

Technical Preliminaries

2.1 Notation

Before formalizing our model and preliminaries, we introduce some convenient notation. For a natural number n, we use $[n]$ to denote the set $\{1, \ldots, n\}$.[1] Given a vector $v = (v_1, \ldots, v_n)$ indexed by $[n]$, where v_i is an abstract element, we use v_{-i} to denote the entries of v other than the ith, and use (v_{-i}, v_i') to denote the result of replacing the ith entry of v with v_i'.

2.2 Mechanism Design Basics

In this section, we review some necessary basics from mechanism design and define our model. For a more comprehensive survey, we refer the reader to [70].

2.2.1 Mechanism Design Problems

An *mechanism design problem* is a family of instances, each of which is given by a *feasible set* Ω, and *valuation functions* $v_1, \ldots, v_n$, where $v_i : \Omega \to \mathbb{R}$ is the valuation function of player i. Elements of Ω are referred to as *outcomes*, and the vector

[1] This is not to be confused with $[a, b]$ for two real numbers a and b, which denotes the set of real numbers between a and b.

$(v_1, \ldots, v_n)$ of valuation functions indexed by players is referred to as a *valuation profile*. Player i's value $v_i(\omega)$ for outcome ω is measured in monetary units.

We consider *direct-revelation mechanisms* for mechanism design problems. For each feasible set Ω and number of players n, such a mechanism comprises an *allocation rule* $\mathcal{A}$, which is a function from (hopefully truthfully) reported valuation functions $v_1, \ldots, v_n$ to an outcome $\omega \in \Omega$, and a *payment rule* p, which is a function from reported valuation functions to a required monetary payment from each player. We allow the allocation and payment rules to be randomized.

We note that the allocation and payment rules are defined in reference to the feasible set Ω and the number of players n, and moreover need only be defined for valuations $v_1, \ldots, v_n$ for which $(\Omega, (v_1, \ldots, v_n))$ is a valid instance of the mechanism design problem. Typical problems we consider are structured, in that not every valuation profile occurs in an instance of the problem. For example, in the combinatorial allocation problem, the valuation function of a particular player must only depend on the bundle that player receives (rather than depending on the bundles received by other players), and must be a non-decreasing set function, among other requirements. Such problems are often said to have *restricted valuations*, and this is true of all problems considered in this thesis. Nevertheless, the general mechanism design problem where valuations are *unrestricted* is of theoretical interest.

2.2.2 The Utility Model

We consider players with *quasilinear utilities* in this thesis. Specifically, a player's *utility* for an execution of a mechanism is defined to be his value for the chosen outcome less his payment. Formally, if outcome ω is chosen by the mechanism and player i is charged p_i, then his utility is equal to $v_i(\omega) - p_i$, where v_i denotes i's valuation function. The employment of money as a yardstick in this manner is known to expand the set of incentive-compatible mechanisms, and is a realistic and mild assumption in many settings. As a result, this utility model is the standard in much of algorithmic mechanism design (see e.g. [70]).

2.2.3 Incentive Compatibility

We recall from Section 2.1 that v_{-i} denotes the valuation of players other than i, and (v_{-i}, v_i') denotes the valuation profile that results from replacing player i's valuation v_i with a different valuation v_i'. We now define the forms of incentive compatibility that we will mention in this thesis. We begin with our main protagonists: A mechanism with allocation and payment rules $\mathcal{A}$ and p is *truthful in expectation* if every player always maximizes his expected utility by truthfully reporting its valuation function, meaning that

$$\mathbf{E}[v_i(\mathcal{A}(v)) - p_i(v)] \geq \mathbf{E}[v_i(\mathcal{A}(v_i', v_{-i})) - p_i(v_i', v_{-i})] \tag{2.1}$$

for every player i, (true) valuation function v_i, (reported) valuation function v_i', and (reported) valuation functions v_{-i} of the other players. The expectation in (2.1) is over the coin flips of the mechanism. If a mechanism $(\mathcal{A}, p)$ is truthful in expectation, we say it *implements* its allocation rule $\mathcal{A}$. If there exists a mechanism implementing allocation rule $\mathcal{A}$, we say that $\mathcal{A}$ is *implementable*.

All mechanisms designed in this thesis are truthful in expectation. These mechanisms incentivize a *risk-neutral* player — i.e., a player who seeks to maximize his expected net monetary payoff — to report his true valuation regardless of the reports of the other players; for such a player, truth-telling is a *dominant strategy*. We note, however, that stronger notions of incentive compatibility are less sensitive to the risk-neutrality assumption, and we mention the main two. If a mechanism satisfies (2.1) for every flip of its internal coins, rather than merely in expectation, we call the mechanism *universally truthful*. If a mechanism is deterministic and satisfies (2.1) then universal truthfulness and truthfulness in expectation coincide, and we refer to it simply as *truthful*. Unfortunately, the space of universally-truthful mechanisms appears to overly constrain design of approximation mechanisms for many problems in algorithmic mechanism design, and more so for deterministic truthful mechanisms, motivating our focus on truthfulness in expectation. We note that we sometimes refer to truthful-in-expectation mechanisms simply as truthful when the qualification is clear from context.

We adopt two additional and standard design goals on top of truthfulness in expectation. A mechanism is *individually rational in expectation* if each player's expected utility on truth-telling is non-negative; formally, $\mathbf{E}[v_i(\mathcal{A}(v)) - p_i(v)] \geq 0$ for each valuation profile v and player i. Individually rationality is also frequently known as *voluntary participation*. For problems where player values are non-negative (i.e. players derive value from the outcomes), as are most considered in this thesis, all our mechanisms will be individually rational in expectation, and each player's expected payment is non-negative on every input. The latter requirement is known as the *non-negative transfers* property. Similarly, for problems where player values are non-positive (i.e., players incur costs from the outcomes), we desire mechanisms that are individually rational in expectation, and where each player's expected payment is non-positive (i.e., players are paid) on every input.[2] We also note that some of our mechanisms satisfy individual rationality and/or the proper direction of payments *universally* — i.e., for each flip of the mechanism's random coins.

2.3 Basics of Optimization and Algorithms

In this section, we define some basic terminology from optimization and approximation algorithms. We consider optimization problems of a general form. An optimization problem Π is a family of instances. Each instance in Π is a pair $I = (\Omega, v)$, where Ω is a *feasible set* of *outcomes*, and $v : \Omega \to \mathbb{R}$ is the objective function. In a *maximization problem*, the goal is to maximize $v(\omega)$ over $\omega \in \Omega$. *Minimization problems* are defined analogously. An optimization problem is *non-negative* if, for all instances (Ω, v), $v(\omega) \geq 0$ for each $\omega \in \Omega$. Most problems we consider in this thesis are non-negative maximization problems, and the remaining are non-negative minimization problems.

An *algorithm* $\mathcal{A}$ for an optimization problem Π takes as input an instance $(\Omega, v) \in \Pi$, and outputs some $\omega \in \Omega$. We allow our algorithms to be randomized. When Π is a non-negative maximization problem and $0 \leq \alpha \leq 1$, we say $\mathcal{A}$ is an α-approximation

[2]However this is not always possible for cost-minimization problems, and we have to relax these requirements for two extensions of our results (Sections 8.6.1 and 10.3.3).

algorithm if $\mathbf{E}[v(\mathcal{A}(\Omega, v))] \geq \alpha \max_{\omega \in \Omega} v(\omega)$ for every instance $(\Omega, v) \in \Pi$; the expectation is taken over the internal random coins of $\mathcal{A}$. Similarly, when Π is a non-negative minimization problem and $\alpha \geq 1$, we say $\mathcal{A}$ is an α-approximation algorithm if $\mathbf{E}[v(\mathcal{A}(\Omega, v))] \leq \alpha \min_{\omega \in \Omega} v(\omega)$ for every instance $(\Omega, v) \in \Pi$. For both maximization and minimization problems, we refer to α and $1/\alpha$ interchangeably as the algorithm's *approximation ratio*. This abuse of notation is for historical reasons and for reasons of convenience, and the intention is always clear from context.[3] We also note that, in general, an approximation ratio α may be a function of the instance (Ω, v).

For a function $t : \Pi \to \mathbb{N}$, we say algorithm $\mathcal{A}$ for Π runs in [expected] time t if it takes at most $t(I)$ steps [in expectation] on each instance $I \in \Pi$. Frequently, t is some polynomial in parameters that describe the instance. For typical optimization problems, where the input is represented explicitly, we say $\mathcal{A}$ runs in [expected] *polynomial time* if it runs in [expected] time t for some polynomial t in the length of representation of the instance. We also consider some problems where inputs are not presented explicitly to the algorithm, but rather can be partially queried during the algorithm's execution. For these problems, polynomial time is defined in reference to natural parameters of the instance other than the length of its explicit representation; when discussing algorithms for such problems, we explicitly mention those parameters unless they are clear from context.[4]

We note that the runtime and approximation ratio of an algorithm, as defined above, are *worst-case* performance guarantees, in that they must hold for *every* instance of the optimization problem at hand.

[3]For example, when discussing a maximization problem and an approximation ratio α greater than 1 is mentioned, it is understood that the corresponding algorithm finds an outcome who's objective value is at least a $1/\alpha$ factor of the maximum objective value over all outcomes in the feasible set.

[4]A typical example is combinatorial allocation, where a valuation function on m items requires 2^m real numbers, one for each bundle of the items, for explicit representation. The natural parameters are taken to be the number of items and number of players, and we consider algorithms that adaptively query valuation functions at specific bundles during their execution.

2.4 Mechanism Design and Optimization

We consider mechanism design problems, as defined in Section 2.2, that are equipped with an *objective function* that is to be maximized or minimized. For each instance $(\Omega, (v_1, \ldots, v_n))$ of a mechanism design problem, an objective function w is a map from Ω to the real numbers. In most of this thesis, we consider the goal of maximizing the *social-welfare* objective: $w(\omega) = \sum_{i=1}^{n} v_i(\omega)$. Other objectives, however, are possible; Chapter 10 considers maximizing a fairness-type objective $w(\omega) = \min_i v_i(\omega)$, among others. We refer to mechanism design problems equipped with an objective function as *mechanism design optimization problems*.

A mechanism design optimization problem doubles as an optimization problem of the form defined in Section 2.3. This allows us to simultaneously explore both computational and incentive-related questions for these problems. In particular, we ask the following kind of question for each problem studied in this thesis: Is there a mechanism for the problem that is (1) incentive compatible, (2) can be implemented in polynomial time, and (3) its allocation rule is an α-approximation algorithm for the corresponding optimization problem.

Finally, we note that it is traditional to refer the outcomes in Ω as *solutions* or *feasible solutions* in an optimization context, and as *allocations* in many mechanism design contexts; we use these terms interchangeably.

2.5 Classifying Mechanism Design Problems

As discussed in Section 2.2, all problems we consider in this thesis have *restricted valuations* — i.e., not all functions mapping the outcomes to the real numbers are valid player valuations. Restricting allowable valuations for a problem expands the sets of incentive-compatible mechanisms, and moreover improves the approximation guarantees of polynomial-time algorithms. Therefore, problems with restricted valuations tend to admit more interesting positive results than their unrestricted counterparts. Given that, additionally, most well-motivated problems impose natural restrictions on player valuations, our focus is justified.

Work in mechanism design theory has evidenced a phase transition in the set of incentive-compatible mechanisms as valuations are restricted. Specifically, incentive compatibility is frequently more permissive for *single-parameter problems*. Informally, an allocation in a single-parameter problem awards each player a real-number "quantity" of a homogeneous good, and each player's valuation is described by a single real number representing his "value per unit of good". A typical example is the single-item allocation problem (Section 1.2.1), where a player's valuation is described by a single real number, namely his value for the item.

We only consider single-parameter problems in Chapter 10, and therefore defer their formal definition to that chapter. Most problems we consider in this thesis require multiple private parameters to describe a player's valuation function, and are therefore called *multi-parameter problems*. A typical example is the combinatorial allocation problem (Section 1.2.2), where a player's valuation function is described by a different private value for each package of items. As described in Chapter 3 (specifically, Section 3.3.2), the space of incentive-compatible mechanisms for many multi-parameter problems appears much more constrained than it is for their single-parameter counterparts.

2.6 Commentary on Our Model

In this section, we place our modeling assumptions and design goals in the proper economic context. We begin by making some of them more explicit. We design *direct-revelation* mechanisms, meaning that the mechanism solicits direct reporting of each player's entire valuation function in a single step. The quasilinear utility model considered in this thesis, defined in Section 2.2.2, assumes that players are *risk neutral* — i.e., a player's utility is linear in his net monetary payoff. In this context, truthful-in-expectation mechanisms are referred to in the microeconomics community as *dominant-strategy incentive-compatible*. We now discuss some of our goals and assumptions in more detail.

Direct revelation. Direct-revelation mechanisms are particularly attractive due to their simplicity: Players report their valuations up-front in a single step, and the mechanism's computation subsequently proceeds "offline". Even though there are settings where indirect mechanisms are more computationally efficient than their direct counterparts (see [22]), direct-revelation mechanisms suffice for the results of this thesis.

A note is in order on the proper interpretation of direct revelation for problems where valuations are accessed via an *oracle model*, as in Chapters 5, 6, and 9. For these problems, valuations are exponential-sized objects, and our mechanisms adaptively query an *oracle* that answers specific questions regarding the valuations. For example, for our result on multi-unit auctions in Chapter 9, there is an exponential number of identical items to be divided between the players, and each player's valuation is presented as an oracle that takes as input a number j of items, and returns the player's value for receiving j items. Our mechanism for multi-unit auctions runs in time much smaller than the number of items, and queries the oracles at specific values during its execution. The oracle model should be viewed as an abstraction of all classes of succinctly-represented valuations for which the corresponding queries to the oracle can be efficiently implemented. An equivalent interpretation is that players report to the mechanism a procedure that answers oracle queries regarding their valuation.[5]

Risk neutrality. The assumption that players are risk neutral is standard in mechanism design, and underlies many of the landmark results of the field (e.g., revenue equivalence theorems [85, 68, 76]). Nevertheless, this assumption has been criticized on the grounds that, in reality, players are frequently *risk averse*.[6] We leave open the general question of whether our results and techniques can be extended to non-risk-neutral environments, though there is reason for optimism: since they were first

[5]It is tempting to interpret a mechanism in an oracle model as an interactive procedure that queries players directly during its execution, allowing players to adaptively choose their reports based on information revealed through the mechanism's choice of queries. This is not our intention.

[6]The most common approach for modeling risk applies a monotonic transformation to our quasi-linear utilities (see [64]). When the transformation is concave this indicates risk aversion, and convexity indicates risk seeking (see [63, Section 6.C]).

published, our results for multi-unit auctions (Chapter 9) and for makespan minimization (Chapter 10) have both been strengthened by follow-up work [86, 20] to universally-truthful mechanisms, which are incentive compatible irrespective of players' attitudes towards risk.[7]

Dominant-strategy incentive compatibility. Dominant-strategy implementation is a *worst-case* notion, much in the analytic tradition of computer science, which in-part explains its adoption as the standard design goal in algorithmic mechanism design. A player facing a dominant-strategy incentive-compatible mechanism is confronted with a sweeping guarantee: regardless of the the preferences, strategies, and rationality of others, truth-telling maximizes his expected utility. As a result, these mechanisms are *robust* to variations in environments in which they are employed, and make no assumptions regarding the beliefs of players about each other. This is a stronger notion than other forms of incentive compatibility considered in mechanism design, most notably *Bayesian incentive compatibility*.

We would be remiss if we did not overview Bayesian incentive compatibility, as it is the dominant paradigm for mechanism design as practiced by the microeconomics community. These mechanisms are defined in reference to a Bayesian environment, where the valuation profile of the players is drawn from a publicly known prior distribution — in technical terms, this distribution is *common knowledge*. A Bayesian incentive-compatible mechanism in such an environment is one where each player maximizes his expected utility by bidding truthfully, *assuming other players' valuations are drawn from the common prior, and moreover are reported truthfully.* The randomness in a player's utility is taken over the random coins of both the mechanism and the draws of other players' valuations. In game-theoretic terms, for these mechanisms truth-telling is a *Bayes-Nash equilibrium*.

In general, a Bayesian incentive-compatible mechanism is not robust to discrepancies between the environment for which it was designed, and that in which it is

[7] We note, however, that incentive compatibility in non-risk-neutral settings appears easier to satisfy than universal truthfulness, assuming standard models such as those in [64].

deployed. Specifically, the incentive compatibility of these mechanisms is contingent on the accuracy of the assumed distribution of player valuations, as well as on the assumption that this distribution is *common knowledge* among the players. Dominant-strategy incentive-compatible mechanisms, on the other hand, are incentive compatible for *every* distribution of player valuations, and regardless of players' beliefs about each other.

Whereas Bayesian incentive compatibility is a weaker guarantee than dominant-strategy incentive-compatibility, it has the advantage of being more permissive. This has enabled stronger positive results in some Bayesian environments, as mentioned in our discussion of related work (specifically, Section 1.7.3).

Chapter 3

Welfare Maximization Background

This chapter serves two goals at once: presenting relevant background and preliminaries for welfare maximization problems, and laying the technical groundwork for our results. Specifically, we motivate the detailed study of welfare maximization problems, compare and contrast known classes of deterministic and randomized mechanisms for approximate welfare maximization, and consequently advocate the class of randomized maximal-in-distributional-range mechanisms as a design framework for these problems.

We begin in Section 3.1 by introducing welfare maximization problems and justifying their central position in this thesis. Section 3.2 describes the welfare-maximizing and truthful Vickrey-Clarke-Groves mechanism, which serves as the technical and conceptual starting point for the design of our approximation mechanisms. We then describe some deterministic mechanisms that generalize VCG in Section 3.3, commenting on their economic and computational significance and limitations. We devote Section 3.4 to *maximal-in-distributional-range (MIDR)* mechanisms; randomized relatives of VCG that are the main protagonists in much of this thesis. Specifically, Section 3.4.1 defines MIDR mechanisms; Section 3.4.2 describes the conceptual realization that these randomized mechanisms can overcome the complexity-theoretic barriers faced by their deterministic counterparts; and Section 3.4.3 proves some formal properties of MIDR mechanisms that we exploit in this thesis.

3.1 Welfare Maximization Problems

A *welfare maximization problem* is a *mechanism design optimization problem*, as defined in Section 2.4, where the objective is to maximize *social welfare*: the sum of all players' values for the allocation. Most of this thesis is concerned with welfare maximization problems, as they occupy center stage in algorithmic mechanism design. We posit three main reasons for their importance:

Welfare maximization problems are pervasive. Social welfare is the canonical objective for a principal interested in the well-being of a group. As a result, it has been considered in many contexts, such as combinatorial auctions, bandwidth allocation, network design, scheduling, and more.

Welfare maximization problems expose computational considerations. Interesting approximation mechanisms, even those that run in exponential time, are rare for multi-parameter mechanism design problems with a non-welfare objective.[1] In contrast, welfare maximization problems admit sweeping positive results in the absence of computational considerations. The truthful Vickrey-Clarke-Groves (VCG) mechanism is optimal (i.e., exactly maximizes welfare) for all these problems. Such a strong and positive result from the economic perspective renders computation the binding constraint in the design of practical mechanisms for these problems. Consequently, these problems expose particularly clearly the conflict between computational efficiency and incentive compatibility, and constitute ripe ground for incorporating computational ideas into mechanism design.

Welfare maximization problems are technically expressive. Objectives other than welfare are common: we mention the principal's *revenue* and various notions of *fairness* in allocation as examples. Whereas resource allocation problems with non-welfare objectives typically require different mechanisms, *useful mechanisms for these*

[1] Whereas few truly positive results for multi-parameter problems with a non-welfare objective are known, some single parameter problems with a non-welfare objective admit useful approximation mechanisms (see Chapter 10 and [3]).

problems often technically resemble those used in welfare maximization. A salient example is revenue maximization for single-parameter problems in the Bayesian setting,[2] where the problem of maximizing revenue subject to incentive compatibility reduces, via a transformation, to the design of a welfare-maximizing incentive-compatible mechanism for the transformed problem (see [68] and [76]). Another example is max-min fairness in scheduling, considered in Chapter 10, where the mechanisms we design are technically similar to the class of maximal-in-distributional-range mechanisms for welfare maximization (described in Section 3.4).

This technical expressiveness of welfare maximization is not merely empirical. It was shown by Roberts [77] that for the general mechanism design problem, where players' values are allowed to be an arbitrary function from allocations to real numbers, the only truthful mechanisms are those that optimize a weighted version of welfare over some subset of the allocations — close relatives of the VCG mechanism that we discuss below. While this theorem ceases to hold formally for many problems that are more structured, such as combinatorial allocation, it appears to hold in spirit for most multi-parameter mechanism design problems. Indeed, useful mechanisms that depart from the VCG family are rare in the literature on multi-parameter problems.[3]

3.2 The Vickrey-Clarke-Groves Mechanism

3.2.1 Definition

We define the Vickrey-Clarke-Groves (VCG) mechanism for a generic mechanism design problem. Fixing the feasible set Ω and the number n of players, the allocation rule $\mathcal{A}$ of VCG simply computes a welfare-maximizing solution; specifically $\mathcal{A}(v_1, \ldots, v_n) \in \text{argmax}_{\omega \in \Omega} \sum_i v_i(\omega)$. The payment rule charges each player i a "pivot" amount $h_i(v_{-i})$ independent of his bid, minus the welfare of the other players

[2] In these environments, player valuations are drawn from a publicly known prior distribution. See Section 2.6 for a brief overview.

[3] There is, however, a small number of interesting non-VCG-based mechanisms for multi-parameter problems in the literature. We mention the mechanisms of [33] for combinatorial auctions as examples.

$\sum_{j \neq i} v_j(\mathcal{A}(v))$. This payment rule renders this deterministic mechanism truthful, and we sketch a proof of this fact in Section 3.2.2.

As defined above, there is flexibility in the choice of the pivot terms in VCG. The "right" choice for many applications is the *Clarke pivot rule*, which sets $h_i(v_{-i}) = \max_{\omega \in \Omega} \sum_{j \neq i} v_j(\omega)$. The payments of VCG are non-negative with this pivot rule, and moreover individual rationality holds when valuations are non-negative-valued functions, and we prove both these properties in Section 3.2.2. Notably, for VCG with the Clarke pivot rule, a player's payment is naturally interpreted as his *externality*: the decrease in welfare of other players due to i's participation in the mechanism.

3.2.2 Proof

For completeness, we now provide a proof sketch of the properties of VCG outlined in Section 3.2.1. We begin with truthfulness. Consider a player i with true valuation v_i, and fix the reports v_j of players j other than i. Since the pivot term $h_i(v_{-i})$ is independent of player i's report, the player may ignore it in maximizing his utility. Therefore, we assume without loss of generality that $h_i(v_{-i}) = 0$. When player i reports b_i, the allocation ω chosen by the mechanism maximizes $b_i(\omega) + \sum_{j \neq i} v_j(\omega)$. Moreover, player i's utility from allocation ω equals his value for the allocation, $v_i(\omega)$, minus his payment, $0 - \sum_{j \neq i} v_j(\omega)$, totaling $v_i(\omega) + \sum_{j \neq i} v_j(\omega)$. Evidently, setting $b_i = v_i$ maximizes player i's utility, as the mechanism essentially "maximizes on the player's behalf".

We now turn our attention to the non-negative transfers and individual-rationality properties when the Clarke pivot rule is used. Observe that player i's payment is equal to the maximum possible welfare of players other than i, less the realized welfare of those players, both according to their reports; this is evidently non-negative. For individual rationality, assume that player i reports his true valuation v_i, and fix the reports v_j of players j other than i. Let ω denote the outcome of the mechanism for these reports. Player i's utility is equal to his value $v_i(\omega)$ for allocation ω, less his payment $\left(\max_{\omega' \in \Omega} \sum_{j \neq i} v_j(\omega')\right) - \sum_{j \neq i} v_j(\omega)$. This can be simplified to $\sum_j v_j(\omega) - \max_{\omega' \in \Omega} \sum_{j \neq i} v_j(\omega')$. Because the mechanism chooses ω to maximize $\sum_j v_j(\omega)$, player

i's utility equals the maximum welfare of all players less the maximum welfare of players other than i, both according to the reported valuations. When player i's value for each allocation is non-negative, the former term is evidently greater than the latter, and consequently player i's utility is non-negative.

3.2.3 Commentary

The Vickrey-Clarke-Groves mechanism, combined with the Clarke pivot rule, is a sweeping positive result from the economic perspective: it is truthful, individually rational, satisfies non-negative transfers, and maximizes welfare. However, since it requires solving the welfare maximization problem optimally, VCG can not be implemented efficiently when the underlying welfare-maximization problem is computationally intractable.

3.3 Deterministic VCG-based Mechanisms

The VCG mechanism is part of a larger family of deterministic mechanisms, known as *affine maximizers*, all of which are incentive compatible. This section defines affine maximizers, starting the most pertinent affine maximizers to this thesis: *maximal-in-range* mechanisms. We then proceed to comment on the significance and limitations of these deterministic generalizations of VCG.

3.3.1 Definitions

Maximal-in-Range Mechanisms

Fix a feasible set Ω and a number of players n. A deterministic allocation rule $\mathcal{A}$ mapping valuation functions $v_1, \ldots, v_n$ to an outcome $\omega \in \Omega$ is *maximal in range* if it maximizes social welfare over a fixed subset of the feasible set Ω, called the *range*. More formally, $\mathcal{A}$ is maximal in range if there exists a subset $\mathcal{R}$ of Ω such that $\mathcal{A}(v_1, \ldots, v_n) \in \text{argmax}_{\omega \in \mathcal{R}} \left(\sum_{i=1}^{n} v_i(\omega)\right)$ for every valuation profile $v_1, \ldots, v_n$.

Observe that the VCG mechanism's allocation rule is the maximal-in-range allocation rule with $\mathcal{R} = \Omega$. An alternative interpretation is that the maximal-in-range allocation rule with range $\mathcal{R}$ is allocation rule of the VCG mechanism when the feasible set is $\mathcal{R}$ rather than Ω. The second interpretation makes it clear that every maximal-in-range allocation rule can be supplemented with payments to yield a truthful mechanism. A *maximal-in-range mechanism* is defined as a truthful mechanism with a maximal-in-range allocation rule.

Affine Maximizers

Fix a feasible set Ω and a number of players n. A deterministic allocation rule $\mathcal{A}$ mapping valuation functions $v_1, \ldots, v_n$ to an outcome $\omega \in \Omega$ is an *affine maximizer* if it maximizes a non-negative affine function of the players' values over a subset of the feasible set Ω. More formally, $\mathcal{A}$ is an affine maximizer if there exists a subset $\mathcal{R}$ of Ω, non-negative per-player weights $\alpha_1, \ldots, \alpha_n$, not all zero, and real-valued per-allocation offsets $\{\beta_\omega\}_{\omega \in \mathcal{R}}$, such that $\mathcal{A}(v_1, \ldots, v_n) \in \text{argmax}_{\omega \in \mathcal{R}} ((\sum_{i=1}^{n} \alpha_i v_i(\omega)) + \beta_\omega)$ for every valuation profile $v_1, \ldots, v_n$.

Observe that the VCG mechanism's allocation rule is the affine maximizer with $\mathcal{R} = \Omega$, all player weights equal to 1, and the per-allocation offsets equal to 0. As in the VCG mechanism, every affine-maximizer allocation rule can be supplemented with payments to yield a truthful mechanism. These payments are related to those for the maximal-in-range allocation rule with range $\mathcal{R}$ by an affine transformation, and the proof is similar. An *affine-maximizer mechanism* is defined as a truthful mechanism with an affine-maximizer allocation rule.

3.3.2 Significance

Generalizations of the VCG mechanism are important for two reasons. First, they are alternatives to the VCG mechanism when welfare maximization is computationally intractable, and in some cases can be used to design incentive-compatible, approximately-optimal, polynomial-time mechanisms for such problems. Second, they

characterize all deterministic truthful mechanisms in some contexts, enabling the proof of impossibility results for deterministic mechanisms.

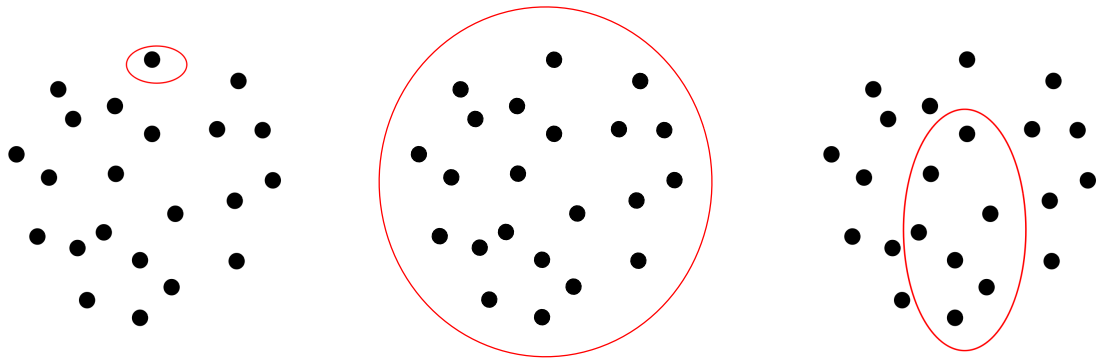
As Approximation Mechanisms

Most importantly, affine maximizers provide an alternative to the intractable VCG mechanism for NP-hard welfare-maximization problems. Maximal-in-range mechanisms, in particular, may simplify the computational task by requiring optimization of welfare only over a subset of the feasible solutions, rather than over the entire set. Affine maximizers that are not maximal in range have historically been less useful for this purpose.[4] The maximal-in-range paradigm reduces what was previously a task of both mechanism design and algorithm design to one that can be viewed as algorithm design in a restricted computational model.

Next, we informally overview the challenges faced in designing computationally-efficient maximal-in-range mechanisms for welfare maximization. First, recall that a maximal-in-range algorithm optimizes social welfare over a subset, called the range, of the feasible allocations. We emphasize that the range is *fixed independently of the player valuations*. For combinatorial allocation, for instance, the range could be the set of allocations of the items where each player receives an even number of items. A more useful example is due to [32], who used a simple range to obtain a nontrivial, yet super-constant, approximation for a variant of combinatorial auctions. Theirs was the range of all allocations where each player gets either all items or a single item (this result was described in Section 1.5.3).

Maximal-in-range algorithms can always be made into a truthful mechanism using VCG payments, as described in Section 3.3.1. This leaves only two requirements: (1) polynomial-time implementability and (2) approximation of the social welfare. For requirement (1), polynomial-time implementation, the range of the maximal-in-range allocation rule should be "small" enough, in some technical sense, so that exact optimization over the range of allocations is possible in polynomial time. For (2),

[4]This is unsurprising, at least for welfare maximization problems. The ability to differentially weight players before learning their valuation functions is useless by a permutation argument. Similarly, per-allocation offsets in affine maximizers are independent of the scale of the valuations, and therefore disappear in the worst-case sense when valuations are much larger.



(a) Small Range (b) Big Range (c) "Sweet Spot" Range

Figure 3.1: Trade-offs in maximal-in-range algorithm design.
Each black point depicts an allocation. In (a),(b), and (c), the range is the circumscribed set of allocations.

approximation of the social welfare, the range should be "large" enough, in the sense that it contains an approximately optimal allocation regardless of the players' valuations. Either of these requirements is easy to satisfy by itself, by choosing a single allocation or all allocations as the range. However, finding a "sweet spot" that satisfies both requirements, or determining if one even exists, is the essence of the challenge in designing truthful polynomial-time mechanisms based on VCG. This trade-off is illustrated in Figure 3.1.

As a Characterization

Affine maximizers are conceptually important in that they capture the power of truthful mechanisms for problems that are at a sufficient level of generality. Concretely, for the mechanism design problem with unrestricted valuations (defined in Section 2.2), Roberts [77] showed that the only deterministic truthful mechanisms are affine maximizers. This *characterization* of truthful mechanisms is evidence that welfare maximization — or, rather, maximization of an affine transformation of welfare over a subset of solutions — is all that is possible to implement via a truthful mechanism for problems at a sufficient level of generality, regardless of computational considerations.

Whereas Roberts' characterization breaks down for many problems that are more structured, such as most of those considered in this thesis, it often continues to hold in spirit for multi-parameter problems: most deterministic mechanisms that have been successfully employed for such problems have been affine maximizers — in fact, maximal in range. Moreover, impossibility results that rule out "good" maximal-in-range mechanisms have on several occasions been accompanied, or followed soon after, by a more general impossibility result for all deterministic truthful mechanisms. We mention two examples. For a variant of combinatorial public projects, Papadimitriou et al. [74] extend Roberts' characterization by showing that all deterministic truthful mechanisms for their problem are affine maximizers, and then show that no affine maximizer can guarantee a good approximation in polynomial time. Another example is in the domain of combinatorial auctions, where Dobzinski and Nisan [29] rule out good approximation mechanisms that are maximal in range, and Dobzinski [26] then extends that result to all deterministic truthful mechanisms via a direct — non-characterization — proof. While such results fuel the working hypothesis that affine maximizers — and in particular maximal-in-range mechanisms — are representative of the power of deterministic truthful mechanisms in many multi-parameter contexts, the extent to which non-affine-maximizers allow the design of better deterministic approximation mechanisms is, in general, still an open question.

3.3.3 Limitations

As described in Section 3.3.2, deterministic variants of VCG, in particular maximal-in-range mechanisms, are the most viable, and for some problems only, approach for the design of deterministic approximation mechanisms. This makes understanding their limitations an important first step to realizing the power of incentive-compatible mechanisms. Moreover, the limitations of these deterministic variants of VCG will motivate and rationalize our use of randomization; we propose *maximal-in-distributional-range mechanisms* as a randomized generalizations of maximal-in-range mechanisms in Section 3.4, and discuss how they side-step some of the limitations of their deterministic counterparts.

We now discuss the limitations of maximal-in-range mechanisms by informally over-viewing a recent impossibility result. The result in question, by Buchfuhrer et al. [17], applies to some variants of combinatorial allocation, and is at least somewhat representative of the various impossibility results for maximal-in-range algorithms in the literature — we mention [74] and [26] as other recent examples. Recall from the discussion of Section 3.3.2 that a maximal-in-range algorithm for combinatorial allocation should optimize welfare over a range of allocations, fixed independently of the valuations, that is both (1) "small" enough for polynomial-time implementation, and (2) "big" enough to guarantee a good approximation. See Figure 3.1. The result in [17], viewed at a high level, shows that these two properties cannot be simultaneously satisfied for the variants of combinatorial allocation they consider, even under a modest approximation requirement.[5]

Their proofs use *VC-dimension* ideas, first used to prove negative results in algorithmic mechanism design in [74]. At a high level, they argue as follows:[6] If a range $\mathcal{R}$ of allocations of m items to n players includes an approximately optimal allocation for each allowable valuation profile, then $\mathcal{R}$ must include *all allocations* of some m' of the items to some n' of the players, where m' and n' are only polynomially smaller than m and n. Since the allocation problem of m' items to n' players is as intractable as that for m items and n players, up to a polynomial factor, this shows that optimization over the range $\mathcal{R}$ requires the solution of an intractable sub-problem. In other words, any range that is "big" enough for a good approximation is also big enough to embed an intractable problem. The "sweet spot" mentioned in Section 3.3.2, and depicted in Figure 3.1, does not exist for the variants of combinatorial allocation they consider.

[5]Their lower bounds rule out maximal-in-range algorithms with an approximation ratio better than $\min(n, \sqrt{m})$, where n denotes the number of players and m denotes the number of items, for many variants of combinatorial allocation.

[6]We note that [17] presents two different impossibility results under different assumptions, and with different bounds. Our description applies only to one of them, chosen for illustrative purposes.

3.4 Randomized VCG-Based Mechanisms

Motivated by the limitations of deterministic VCG-based mechanisms, discussed in Section 3.3.3, we advocate the design of randomized approximation mechanisms for welfare maximization. This section defines *maximal-in-distributional-range* mechanisms, randomized analogues of the maximal-in-range mechanisms of Section 3.3.1, and presents some intuition as to how they overcome the limitations of their deterministic counterparts. We also establish some formal properties of maximal-in-distributional-range mechanisms that we exploit in the remainder of this thesis.

3.4.1 Definition

Fix a feasible set Ω and a number of players n. Let $dist(\Omega)$ denote the set of probability distributions over Ω, and let $\mathcal{R} \subseteq dist(\Omega)$ be a compact subset of them. The corresponding *maximal-in-distributional-range (MIDR)* allocation rule is defined as follows: given reported valuation functions $v_1, \ldots, v_n$, return an outcome that is sampled randomly from a distribution $D^* \in \mathcal{R}$ that maximizes the expected welfare $\mathbf{E}_{\omega \sim D}[\sum_i v_i(\omega)]$ over all distributions $D \in \mathcal{R}$. Naturally, we refer to $\mathcal{R}$ as the *distributional range* of the allocation rule, or simply the *range* when the qualification is clear from context. We illustrate a distributional range in Figure 3.2.

Observe that a maximal-in-range algorithm is a maximal-in-distributional-range algorithm whose range $\mathcal{R}$ consists only of point distributions. Adopting a different perspective, the MIDR allocation rule is simply that of the VCG mechanism applied to a different mechanism design problem: the problem with feasible set $\mathcal{R}$, and players with values for a lottery equal to their expected values over individual draws from the lottery. The second interpretation makes it clear that every maximal-in-distributional-range allocation rule can be supplemented with payments to yield a truthful-in-expectation mechanism. A *maximal-in-distributional-range mechanism* is defined as a truthful-in-expectation mechanism with a maximal-in-distributional-range allocation rule.

We now examine payments for MIDR mechanisms more closely. Consider an MIDR allocation rule $\mathcal{A}$ with range $\mathcal{R}$. The following adaptation of VCG payments

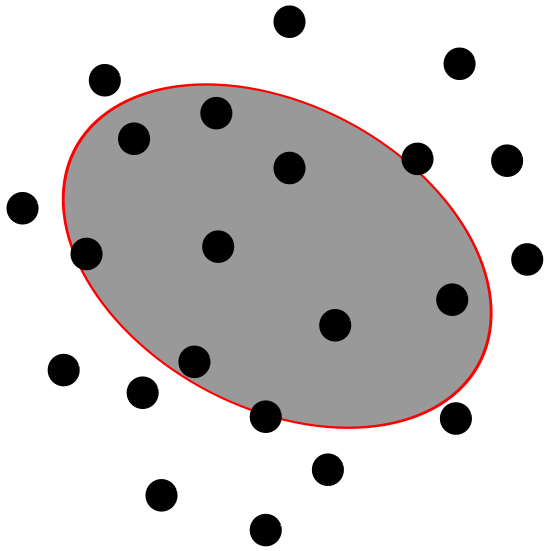


Figure 3.2: A distributional range.
Each black point depicts an allocation. The range is a family of lotteries over allocations, depicted as the gray region inside the convex hull of allocations.

with the Clarke pivot rule is deterministic, and yields a truthful-in-expectation mechanism when combined with $\mathcal{A}$:

$$p_i^{vcg}(v) = \max_{D \in \mathcal{R}} \mathop{\mathbf{E}}_{\omega \sim D} \left[\sum_{i' \neq i} v_{i'}(\omega) \right] - \mathop{\mathbf{E}}_{\omega \sim \mathcal{A}(v)} \left[\sum_{i' \neq i} v_{i'}(\omega) \right]. \tag{3.1}$$

Unfortunately, it is not always possible to compute these VCG payments efficiently, even if $\mathcal{A}$ is implemented efficiently. Luckily, we have additional flexibility: any (randomized) payment rule p satisfying $\mathbf{E}[p_i(v)] = p_i^{vcg}(v)$ also guarantees truthfulness in expectation. We show in Section 3.4.3 that, in most cases, such truth-telling payments p can be computed using only black-box access to $\mathcal{A}$, incurring only polynomial overhead in runtime.

Finally, we note that *distributional affine maximizers* can be defined as a generalization of maximal-in-distributional-range mechanisms. This is analogous to the definition of affine maximizers as a generalization of maximal-in-range mechanisms in Section 3.3. However, like their deterministic counterparts, distributional affine

maximizers that are not maximal in distributional range will not be of much use in this thesis.

3.4.2 Significance

As described in Section 3.3, deterministic variants of VCG are the most viable approach for the design of deterministic approximation mechanisms for welfare maximization. Yet, as described in Section 3.3.3, impossibility results have exposed the limitations of the maximal-in-range computational model. These limitations are frequently due to the non-existence of a "good" range of allocations: A set of allocations is either too "complex" for polynomial-time welfare maximization, or too "small" to guarantee a good approximation of social welfare for an unknown valuation profile.

We observe that this limitation is in-part due to the discrete nature of deterministic ranges. Optimization over discrete sets is frequently NP-hard, whereas continuous and convex optimization problems are frequently more tractable. This contrast in complexity between discrete and continuous problems is the rationale for maximal-in-distributional-range algorithms: In settings where no "good" range of deterministic allocations exists, a range of lotteries may adequately approximate the space of allocations while remaining "simple" enough for polynomial-time exact optimization. In other words, distributional ranges enable combining "bigness" with "simplicity". This phenomenon is depicted in Figure 1.2.

A note is in order on the similarities and differences between designing distributional ranges and designing fractional relaxations for discrete problems. In one respect, the two tasks are similar: one seeks a continuous problem that adequately approximates a discrete one, and yet is more tractable to solve. However, there is an important difference that magnifies the challenge in designing a distributional range: the continuous problem must correspond, at each point in its feasible set, to a lottery over the feasible set of the original discrete problem. Unlike, say, a typical linear programming relaxation, a distributional range must have *no integrality gap*. Whereas a linear program is typically used as a *fractional relaxation* of a discrete problem, a distributional range can analogously be seen as a *fractional contraction*.

3.4.3 Properties

Payment Computation in Polynomial Time

In most cases, truth-telling payments for an MIDR allocation rule $\mathcal{A}$ can be computed using only black-box access to $\mathcal{A}$, incurring only polynomial overhead in runtime. There is one requirement, which facilitates computation of the Clarke pivots: the zero function should be a valid valuation for each player. Formally, we say a problem Π *allows zero valuations* if, whenever $(\Omega, (v_1, \ldots, v_n))$ is a valid instance of Π, and $\mathbf{0}$ denotes the valuation function mapping Ω identically to 0, then for each i the pair $(\Omega, (v_{-i}, \mathbf{0}))$ is also a valid instance of Π. We summarize payment computation for MIDR mechanisms in Proposition 3.4.1. We note that, as in the VCG mechanism, individually rationality is contingent on non-negative valuations.

Proposition 3.4.1. *Fix a mechanism design problem that allows zero valuations. Fix the feasible set Ω, let $\mathcal{A}$ be an MIDR allocation rule with distributional range $\mathcal{R}$, and let p^{vcg} be the VCG payments for $\mathcal{A}$ as in Equation* (3.1)*. There exists a randomized payment rule p such that $\mathbf{E}[p_i(v)] = p_i^{vcg}(v)$, and moreover p can be implemented in* poly(n) *time given black-box access to $\mathcal{A}$. The resulting mechanism $(\mathcal{A}, p)$ is truthful in expectation, and its payments are non-negative in expectation. Moreover, $(\mathcal{A}, p)$ is individually rational in expectation when valuations are non-negative-valued functions.*

Proof. Recall from Section 3.2 that VCG with the Clarke pivot rule is truthful, its payments are non-negative, and it satisfies individual rationality when valuations are non-negative. By linearity of expectations, the analogue of the VCG mechanism that samples a welfare-maximizing lottery in $\mathcal{R}$ and charges expected externalities p^{vcg} satisfies the same properties in expectation. Moreover, it is evident that the same holds for any payment rule p with $\mathbf{E}[p_i(v)] = p_i^{vcg}(v)$ for each i and v.

To complete the proof, we show how to compute such a payment rule p using only a polynomial number of invocations of $\mathcal{A}$ as a black box. We sample random variable $p_i(v)$ as follows: Let x be a sample from lottery $\mathcal{A}(v)$, let x_{-i} be a sample from lottery $\mathcal{A}(v_{-i}, 0)$, and let $p_i(v) = \sum_{i' \neq i} v_{i'}(x_{-i}) - \sum_{i' \neq i} v_{i'}(x)$. Using Equation (3.1) and the fact that $\mathcal{A}$ is MIDR with range $\mathcal{R}$, we conclude that $\mathbf{E}[p_i(v)] = p_i^{vcg}(v)$ for each i and v. This completes the proof. □

Closure Under Random Composition

As our second property of MIDR algorithms, we prove that they are closed under random composition, in the following sense: An algorithm that flips a random coin, and based on the outcome of the coin decides on an MIDR algorithm to run, is also an MIDR algorithm.

Lemma 3.4.2. *An allocation rule that chooses an MIDR allocation rule randomly from an arbitrary distribution over such rules is also an MIDR allocation rule.*

Proof. We fix a feasible set Ω and consider an allocation rule $\mathcal{A}$ that randomly picks an MIDR allocation rule to run. We assume that $\mathcal{A}$ runs the MIDR allocation rule $\mathcal{A}_k$ with probability p_k, and use $\mathcal{R}_k$ to denote the distributional range of $\mathcal{A}_k$. The distributional range of $\mathcal{A}$ is, therefore, a subset of

$$\mathcal{R} = \left\{ \sum_k p_k D_k : D_k \in \mathcal{R}_k \right\},$$

where $\sum_k p_k D_k$ denotes the distribution that samples from distribution D_k with probability p_k.

Fix a valuation profile $v = (v_1, \ldots, v_n)$, and let D_k^v be the distribution of $\mathcal{A}_k(v)$. By definition, $\mathcal{D}_k^v \in \operatorname{argmax}_{D \in \mathcal{R}_k} \{\mathbf{E}_{\omega \sim D}[\sum_i v_i(\omega)]\}$. Now, the distribution of $\mathcal{A}(v)$ can be written as $D^v = \sum_k p_k D_k^v$. Since D_k^v maximizes expected welfare over all elements of $\mathcal{R}_k$ for every k, we conclude that D^v maximizes expected welfare over $\mathcal{R}$. Therefore, $\mathcal{A}$ is an MIDR allocation rule. □

Part II

Convex Rounding

Chapter 4

The Convex Rounding Framework

4.1 Introduction

In this chapter we introduce a new framework for designing approximation mechanisms for welfare maximization problems, based on randomized rounding algorithms and convex optimization. Our framework can be viewed as an adaptation of the dominant approach for the design of (non-truthful) approximation algorithms, that of mathematical relaxation and randomized rounding, to the design of incentive-compatible mechanisms. A typical such algorithm first optimizes over a fractional relaxation of the original problem, and then randomly rounds the fractional solution to an integral one. With rare exceptions, such algorithms cannot be converted into truthful mechanisms.

The high-level idea of our framework is to optimize *directly over the (random) output of the rounding algorithm*, rather than over the *input* to the rounding algorithm. This approach results in maximal-in-distributional-range approximation algorithms, implementable as part of a truthful-in-expectation mechanism. Moreover, when the rounding algorithm results in a concave objective function in the corresponding optimization problem, in which case we refer to it as a *convex rounding algorithm*, the mechanism can be implemented efficiently using techniques from convex optimization.

4.1.1 Summary of Results and Techniques

The most common paradigm for designing approximation algorithms is based on first *relaxing* the problem to a continuous optimization problem (frequently a linear program), *solving* the relaxation, then *rounding* the fractional solution to an integral one via a possibly randomized procedure. When application of the rounding algorithm decreases the objective value of the fractional solution by a factor of at most α, in which case we say the rounding algorithm is α-approximate, the result is an α-approximation algorithm. Almost all randomized rounding algorithms in the literature that follow this approach are not maximal in distributional range, and rarely can they be used as part of an incentive-compatible mechanism.

We propose an adaptation of this paradigm that yields MIDR approximation algorithms. Informally, our main result reduces the design of a truthful-in-expectation mechanism for a particular problem to the design of a rounding algorithm for a mathematical relaxation of that problem satisfying an additional property, which we call *convexity*. When the rounding algorithm is additionally α-approximate, the allocation rule of the resulting mechanism is an α-approximation algorithm.

The high-level idea behind our approach is to optimize *directly on the outcome of the rounding algorithm*, rather than merely on the outcome of the relaxation algorithm (the *input* to the rounding algorithm). In other words, let $(\Omega, \{v_i\}_{i=1}^n)$ be an instance of a mechanism design problem, and let $\mathcal{P} \subseteq \mathbb{R}^m$ be a mathematical relaxation of feasible set Ω. Let r denote a randomized rounding algorithm mapping fractional allocations in $\mathcal{P}$ to integer allocations in Ω. Given players' valuations $v_1, \ldots, v_n$, we compute a fractional allocation $x \in \mathcal{P}$ that maximizes the expected welfare $\mathbf{E}_{\omega \sim r(x)}[\sum_i v_i(\omega)]$ over all fractional allocations x. This methodology evidently gives MIDR algorithms; the distributional range is simply the range of the randomized rounding algorithm r. This optimization problem is often intractable, but when the rounding algorithm r and the space of valuations v are such that the function $\mathbf{E}_{\omega \sim r(x)}[\sum_i v_i(\omega)]$ is always concave in x — in which case we call r a *convex rounding algorithm* — it can typically be solved in polynomial time using convex programming.

4.1.2 Related Work

Both technically and conceptually, the starting point for convex rounding is the linear-programming-based framework of Lavi and Swamy [60]. In fact, our framework can be viewed as relaxing some of the requirements of theirs. We briefly recap their approach in a manner that, while different from their original presentation, facilitates comparison to ours.

Lavi and Swamy identify a class of welfare-maximization mechanism design problems and linear programming relaxations of them with a remarkable property: There is a polynomial-time rounding algorithm such that the expected welfare of rounding a point x of the linear program is equal to the objective value of the linear program at x, scaled down by the linear program's *integrality gap*[1]. The problem of maximizing welfare over the output of this rounding algorithm is equivalent, up to the scaling factor, to solving the linear program, which is typically possible in polynomial time. The result is a maximal-in-distributional-range allocation rule with an approximation ratio equal to the linear program's integrality gap. Perhaps surprisingly, they show that there are interesting problems, such as welfare maximization in combinatorial auctions with unrestricted valuations, that admit relaxations and rounding schemes that lead to the best-possible approximation guarantee when employed in this manner (assuming $P \neq NP$). However, their requirements on the rounding algorithm leave many problems, such as combinatorial auctions and public projects with restricted valuations, out of reach.

Our framework relaxes the — in retrospect often unnecessary — requirement that the expected welfare of rounding is a scaled copy of the objective function of the linear program. We allow the expected welfare of rounding a fractional point x to be an arbitrary concave function of the variables of x. This enriches the space of rounding algorithms that we may design. It is is this flexibility that enables our results for combinatorial auctions (Chapter 5) and public projects (Chapter 6) with restricted valuations — problems that have eluded prior techniques despite intense study.

[1]The integrality gap of a linear program is the maximum, over all linear objectives, of the ratio of the objective's maximum value in the program's feasible region to its maximum value over integer points in the feasible region.

We also single out the work of Dobzinski, Fu, and Kleinberg [28] on combinatorial auctions as a precursor to our framework. The authors designate a particular rounding algorithm, and define a set of "proxy bidders" such that optimizing with respect to these proxy bidders is equivalent to optimizing over the output of their randomized rounding algorithm.

4.2 Relaxations and Rounding Schemes

Let Π be an optimization problem, as defined in Section 2.3. An instance (Ω, v) of Π is described by the following mathematical program.

$$\begin{aligned} &\text{maximize} && v(x) \\ &\text{subject to} && x \in \Omega. \end{aligned} \tag{4.1}$$

We assume Ω is encoded as a subset of some Euclidean space $\mathbb{R}^m$. A *relaxation* Π' of Π defines for every $(\Omega, v) \in \Pi$ a convex and compact *relaxed feasible set* $\mathcal{P} \subseteq \mathbb{R}^m$ that is independent of the objective v (we suppress the dependence on Ω) and satisfies $\Omega \subseteq \mathcal{P}$; and an *extension* $v^{\mathcal{P}} : \mathcal{P} \to \mathbb{R}$ of objective v to the relaxed set $\mathcal{P}$. This gives the following *relaxed mathematical program.*

$$\begin{aligned} &\text{maximize} && v^{\mathcal{P}}(x) \\ &\text{subject to} && x \in \mathcal{P}. \end{aligned} \tag{4.2}$$

Generally, the extension is defined so that it is computationally tractable to find a point $x \in \mathcal{P}$ that maximizes $v^{\mathcal{P}}(x)$ (possibly approximately).

For example, Ω could be the allocations of m items to n bidders in a combinatorial auction (See Chapter 5), $v(x)$ the welfare of an allocation x, $\mathcal{P}$ the feasible region of a linear programming relaxation, and $v^{\mathcal{P}}$ a natural — perhaps linear — extension of v to fractional allocations.

The optimal solution of the relaxed problem (4.2) need not be in Ω. A *rounding scheme* for relaxation Π' of Π defines for each feasible set Ω of Π, and its corresponding relaxed set $\mathcal{P}$, a (possibly randomized) function r from $\mathcal{P}$ to Ω. Since our rounding algorithms will be randomized, we let $dist(\Omega)$ denote the distributions supported on

Ω, and think of r as a function from $\mathcal{P}$ to $dist(\Omega)$. Commonly, the rounding scheme satisfies the following approximation guarantee: $\mathbf{E}_{y\sim r(x)}[v(y)] \geq \alpha \cdot v^{\mathcal{P}}(x)$ for every $x \in \mathcal{P}$. In this case, if x^* maximizes $v^{\mathcal{P}}$ over $\mathcal{P}$ and $v^{\mathcal{P}}$ agrees with v on Ω, then $\mathbf{E}_{y\sim r(x^*)}[v(y)] \geq \alpha \cdot \max_{y\in\Omega} v(y)$.

4.3 Convex Rounding Schemes

We consider optimization problems where the objective function v is the social welfare. Our technique is motivated by the following observation: instead of solving the relaxed problem and subsequently rounding the solution, why not *optimize directly on the outcome of the rounding scheme*? In particular, consider the following relaxation of Π that "absorbs" rounding scheme r into the objective.

$$\begin{array}{ll} \text{maximize} & \mathbf{E}_{y\sim r(x)}[v(y)] \\ \text{subject to} & x \in \mathcal{P}. \end{array} \tag{4.3}$$

The solution to this problem rounds to the best possible distribution in the range of the rounding scheme, over all possible fractional solutions in $\mathcal{P}$. We depict the feasible set of this optimization problem in Figure 1.4. While this problem is often intractable, it always leads to an MIDR allocation rule.

Parameter: Feasible set Ω of Π.
Parameter: Relaxed feasible set $\mathcal{P} \subseteq \mathbb{R}^m$.
Parameter: (Randomized) rounding scheme $r : \mathcal{P} \to dist(\Omega)$.
Input: Objective $v : \Omega \to \mathbb{R}$ satisfying $(\Omega, v) \in \Pi$.
Output: Feasible solution $z \in \Omega$.
1: Let x^* maximize $\mathbf{E}_{y\sim r(x)}[v(y)]$ over $x \in \mathcal{P}$.
2: Let $z \sim r(x^*)$

Algorithm 4.1: Allocation rule optimizing over the range of a rounding scheme.

Lemma 4.3.1. *Algorithm 4.1 is an MIDR allocation rule.*

We say a rounding scheme r is α-*approximate* for $\alpha \leq 1$ if $v(x) \geq \mathbf{E}_{y\sim r(x)}[v(y)] \geq \alpha \cdot v(x)$ for every $x \in \Omega$. When r is α-approximate, so is the allocation rule of Algorithm 4.1.

Lemma 4.3.2. *If r is an α-approximate rounding scheme, then Algorithm 4.1 returns an α-approximate solution (in expectation) to the original optimization problem 4.1.*

For most rounding schemes in the approximation algorithms literature, the optimization problem (4.3) cannot be solved in polynomial time (assuming $P \neq NP$). There is a simple reason for this phenomenon: If rounding scheme r always rounds a feasible solution to itself – i.e., $r(x) = x$ for all $x \in \Omega$ — then an optimal solution to (4.3) is also optimal for (4.1). Thus, in this case, hardness of the original problem (4.1) implies hardness of (4.3). We conclude that we need to design rounding schemes with the unusual property that $r(x) \neq x$ for some $x \in \Omega$.

We now define the class of rounding schemes we use in our results. We call a (randomized) rounding scheme $r : \mathcal{P} \to dist(\Omega)$ *convex* if $\mathbf{E}_{y\sim r(x)}[v(y)]$ is a concave function of $x \in \mathcal{P}$. Our results are possible because convex rounding schemes result in convex programs, and convex programs are typically solvable in polynomial time.

Lemma 4.3.3. *When r is a convex rounding scheme for Π',* (4.3) *is a convex program.*

Under additional technical conditions, discussed in the context of combinatorial auctions in Appendix B.1 and combinatorial public projects in Appendix B.2, the convex program (4.3) can be solved efficiently (e.g., using the ellipsoid method).

The preceding discussion reduces the design of a polynomial-time α-approximate truthful-in-expectation mechanism to the design of a polynomial-time α-approximate convex rounding scheme, modulo some technical conditions. In summary, Lemmas 4.3.1, 4.3.2, and 4.3.3, combined with Proposition 3.4.1, give the following informal theorem.

Theorem 4.3.4. *(Informal) Let Π be a welfare-maximization optimization problem, and let Π' be a relaxation of Π. If there exists a polynomial-time, α-approximate, convex rounding scheme for Π', then there exists a truthful-in-expectation, polynomial-time, α-approximate mechanism for Π.*

Of course, there is no reason a priori to believe that useful convex rounding schemes – let alone ones computable in polynomial time – exist for any important problems. We show in Chapters 5 and 6 that they do in fact exist, and yield new results for two well-studied problems in algorithmic mechanism design.

Chapter 5

Combinatorial Auctions

5.1 Introduction

In this chapter we consider welfare maximization in combinatorial auctions, where a set of non-identical items must be allocated among competing players, each of whom is equipped with private preferences over "packages" of the items. Combinatorial auctions occupy a central position in literature on mechanism design and its application, and have been applied or proposed in many practical contexts.

We use the convex rounding framework introduced in Chapter 4 to design an expected-polynomial-time, truthful-in-expectation, $(1 - 1/e)$-approximation mechanism for welfare maximization in a fundamental class of combinatorial auctions. Our approximation factor is the best possible for the class of combinatorial auctions we consider, assuming $P \neq NP$. Ours is the first truthful-in-expectation and polynomial-time mechanism to achieve a constant-factor approximation for an NP-hard welfare maximization problem in combinatorial auctions with heterogeneous goods.

5.1.1 Summary of Results and Techniques

In *combinatorial auctions*, there is a set of items up for sale, and a set of self-interested players each equipped with a private valuation over subsets of the items. The valuation functions must be non-decreasing (adding items to a bundle does not decrease

its value) and normalized (the value for the empty bundle of items is 0), and may be additionally *restricted* in variants of the problem. The principal is an auctioneer who must allocate the items among the players. The auctioneer's goal is to maximize social welfare, which in this context is the sum of the players' values for the bundles they receive.

Combinatorial auctions enjoy paradigmatic status among resource allocation problems. We quote Blumrosen and Nisan [14]: "combinatorial auctions serve as a common abstraction for many resource allocation problems in decentralized computerized systems such as the Internet, and may serve as a central building block of future electronic commerce systems." It is therefore unsurprising that combinatorial auctions have already been applied or considered in many contexts, such as in allocation of electromagnetic spectrum, allocation of airport take-off and landing slots, and more.

In many existing and potential applications of combinatorial auctions, the welfare-maximizing VCG mechanism can not be deployed, due in part to the computational intractability of the welfare maximization problem. Combinatorial auctions that are actually employed, for instance in the practical settings mentioned above, are often heuristic in nature, specifically tailored to particular markets, and rigorous guarantees on their performance are rare in all but the simplest of settings. This has motivated a major research direction in algorithmic mechanism design, seeking a rigorous understanding of the space of computationally-efficient mechanisms for variants of combinatorial auctions. Despite intense study over the past decade, however, constant-factor approximation mechanisms for combinatorial auctions have eluded researchers, even for variants of the problem where constant-factor approximation algorithms are known. The most studied such variant assumes that player valuations are *submodular*.[1]

Using the convex rounding framework introduced in Chapter 4, we design an expected-polynomial-time, truthful-in-expectation, $(1 - 1/e)$-approximation mechanism for welfare maximization in combinatorial auctions, when players have valuations that are *matroid rank sums (MRS)*. Matroid rank sum valuations encompass most

[1]Submodular functions are those set functions v that satisfy *diminishing marginal returns*: specifically, the marginal value $v(S \cup \{j\}) - v_i(S)$ for a each fixed item j is non-increasing in S.

concrete examples of submodular functions studied in this context, including coverage functions, matroid weighted-rank functions, and convex combinations thereof. Our approximation factor is the best possible, even for known and explicitly given coverage valuations, assuming $P \neq NP$.

To prove the main result of this chapter, we design a convex rounding scheme for combinatorial auctions. Our *Poisson rounding scheme* assigns items randomly and independently, using probabilities that correspond to a distortion of variables from the fractional solution via the map $x \to 1 - e^{-x}$. We exploit the structure of MRS valuations to prove our scheme convex, using ideas from matroid theory and convex analysis.

5.1.2 Related Work

Combinatorial auctions have been studied by researchers in many disciplines, and the literature on them is vast both in its depth and in the breadth of the design constraints and environments considered. We can only hope to provide a glimpse of this body of work, biased as it may be to placing our work in the proper context. We begin with a brief outline of some practical applications of combinatorial auctions, and then proceed to overview existing work in the design of approximation mechanisms for variants of the problem. For a more comprehensive overview of research on combinatorial auctions, we refer the reader to references by Cramton et al. [23] and Milgrom [65], and a survey by Blumrosen and Nisan [13].

The Practice of Combinatorial Auctions

Combinatorial auctions have been considered and/or implemented in many contexts. The most prominent application is to allocation of the electromagnetic spectrum: many spectrum auctions have been employed by governments around the world since the mid 1990s, resulting in the sale of hundreds of billions of dollars worth of spectrum licenses (see [65]). Other applications include airport take-off and landing slot allocation, bus route allocation, and industrial procurement (see e.g. [23]).

The VCG mechanism is not employed, or even considered seriously, for many of the applications mentioned in the preceding discussion. This is in-part due to the computational intractability of its allocation rule, among other critiques.[2] Work in algorithmic mechanism design, like the results of this chapter, isolates the first critique of the welfare-maximizing auction, and proposes the design of polynomial-time mechanisms with rigorous approximation guarantees as an alternative. This contrasts with much of the remaining literature and existing implementations of combinatorial auctions, which forgo rigorous guarantees on the quality of the allocation in favor of auctions with other desirable properties, such as simplicity, resistance to collusion, and empirical performance in specific markets. (See [14], [13], and [65] for a discussion).

Approximation Mechanisms for Combinatorial Auctions

As we describe in Section 5.2.2, variants of combinatorial auctions are described via a class of valuations and an oracle model describing how these valuations are accessed. Many combinations of valuation class and oracle model have been considered, and we refer the interested reader to [13, Figure 11.2] for an overview of known positive and negative results for these variations. The most notable fact from prior work to the work of this chapter, however, can be summarized easily: No constant-factor approximation mechanisms for combinatorial auctions with heterogeneous goods have been obtained, even in settings where constant-factor approximation algorithms exist. The most promising and natural class of valuations for which such a positive result has been pursued is the class of submodular valuations.

Despite intense study prior to the work presented in this chapter, there were no truthful-in-expectation and polynomial-time constant-factor approximation mechanisms for welfare maximization with any non-trivial subclass of submodular bidder valuations. The best previous results, which apply to all submodular valuations,

[2]The VCG mechanism has other weaknesses that have been criticized in a combinatorial auctions context, including vulnerability to collusion, low seller revenues in some settings, and more. See Ausubel and Milgrom [5] for a discussion.

are a truthful-in-expectation $O\left(\frac{\log m}{\log \log m}\right)$-approximation mechanism in the communication complexity model due to Dobzinski et al. [28], and a universally-truthful $O(\log m \log \log m)$-approximation mechanism in the demand oracle model due to Dobzinski [25].

A series of works have provided evidence that computational efficiency and universal truthfulness are in conflict for combinatorial auctions. Universally-truthful, polynomial-time, VCG-based mechanisms with constant approximation ratios have been ruled out for submodular combinatorial auctions in the communication complexity model [29], as well as in the computational complexity model for some explicitly represented sub-classes of submodular functions [17]. In recent work, Dobzinski [26] proved that, in the value oracle model, there is no universally-truthful and polynomial-time mechanism for submodular combinatorial auctions achieving an approximation ratio better than $m^{\epsilon-1/2}$. These results suggest that our relaxation to truthfulness in expectation may be essential for our result.

The result of this chapter presented promise for the design of constant-factor approximation mechanisms for more general variants of combinatorial auctions. This was recently shown to be impossible in the value oracle model in work by Dughmi and Vondrák [38], who proved that there is no truthful-in-expectation and polynomial-time mechanism for submodular combinatorial auctions achieving an approximation ratio better than m^{γ}, for some specific constant γ. Due to the similarity between the lottery-value oracles of this chapter and traditional value oracles, this latest result is evidence that the results of this chapter are unlikely to be extended to submodular valuations without additional assumptions. Specifically, it is reasonable to conjecture that matroid rank sum valuations are near the limit of what can be handled by a polynomial-time constant-factor approximation mechanism for combinatorial auctions.

5.2 Model and Preliminaries

5.2.1 Combinatorial Auctions

In *combinatorial auctions* there is a set $[m] = \{1, 2, \ldots, m\}$ of items, and a set $[n] = \{1, 2, \ldots, n\}$ of players. Each player i has a valuation function $v_i : 2^{[m]} \to \mathbb{R}_+$ that is normalized ($v_i(\emptyset) = 0$) and non-decreasing ($v_i(A) \leq v_i(B)$ whenever $A \subseteq B$). A feasible solution is an *allocation* $(S_1, \ldots, S_n)$, where S_i denotes the items assigned to player i, and $\{S_i\}_i$ are mutually disjoint subsets of $[m]$. Player i's value for outcome $(S_1, \ldots, S_n)$ is equal to $v_i(S_i)$. We consider combinatorial auctions when the goal is to find an allocation $(S_1, \ldots, S_n)$ that maximizes the *social welfare*: $\sum_i v_i(S_i)$.

For the main result of this chapter, we assume that each player's valuation function is a *matroid rank sum (MRS)* function (Section 5.2.3), presented via a *lottery-value oracle* (Section 5.2.4).

5.2.2 Describing Variants of Combinatorial Auctions

Variants of combinatorial auctions are typically specified by two parameters: A *class of valuations* assumed to include each player's valuation function, and a corresponding *oracle model* that describes how the algorithm may access the valuation. As an example, the least tractable class of valuations typically considered, often referred to as the set of *unrestricted valuations*,[3] is the family of all functions from bundles to real numbers that are non-decreasing and normalized. Combinatorial auctions are most interesting to study when additional structure is assumed of the valuation functions, in which case we say the combinatorial auctions problem has *restricted valuations*.

[3]Note that this term is overloaded. In context of a general mechanism design problem, an unrestricted valuation may depend arbitrarily on the allocation. On the other hand, an unrestricted valuation for combinatorial auctions must depend only on the player's own bundle, and be normalized and non-decreasing in said bundle, but may otherwise be arbitrary.

Restricted classes of valuations that have been considered include the previously described class of submodular valuations,[4] *subadditive valuations*,[5] *single-minded valuations*,[6] and others. Our main result applies to *matroid rank sum valuations*, a large sub-class of submodular valuations that we define in Section 5.2.3.

In general, a valuation function on m items is an exponential-sized object, requiring 2^m real numbers for explicit representation. This motivates various *oracle models* as alternatives to explicit representation. The most traditional such model is that of *value oracles*, where an algorithm may query a valuation function at any specific bundle in constant time. A more powerful model is that of *demand oracles*, where a valuation v is given by an oracle that takes as input a price p_j for each item j, and returns the bundle S maximizing the player's utility $v(S) - \sum_{j \in S} p_j$.[7] The most powerful access model of all is the *communication complexity model*, which assumes that an oracle for each valuation v answers arbitrary questions that can be formulated and answered using a polynomial number of bits.

It is also not uncommon to directly consider combinatorial auctions with valuations that are represented explicitly, rather than via an oracle. Valuation classes considered in this manner are typically defined in reference to a particular short representation; examples include single-minded valuations, and the class of coverage valuations described in Section 1.2.2 and studied in Section 5.6.1. When valuations are explicitly represented, the mechanism's runtime is allowed to depend polynomially on the size of the representation, as is traditional in algorithm design. This obviates the need for an oracle model, and allows traditional complexity-theoretic analysis of the resulting combinatorial allocation problem. On the other hand, the main advantage of an oracle model is that it enables positive results that make minimal assumptions on how valuations are represented. For example, a polynomial-time algorithm or mechanism for submodular combinatorial auctions in the value oracle

[4]Recall that submodular functions are those set functions v that satisfy *diminishing marginal returns*: specifically, the marginal value $v(S \cup \{j\}) - v_i(S)$ for a each fixed item j is non-increasing in S.

[5]A subadditive valuation $v : 2^{[}m] \to \mathbb{R}$, also known as *complement free*, satisfies $v(A) + v(B) \geq v(A \cup B)$ for all bundles $A, B \subseteq [m]$. Subadditive valuations strictly include submodular valuations.

[6]Single-minded valuations are those that evaluate to 0 for all but a single bundle.

[7]It was shown in [12] that demand oracles can be used to efficiently implement value oracles.

model applies to any explicitly represented subclass of submodular valuations for which value queries can be computed efficiently.

5.2.3 Matroid Rank Sum Valuations

We now define matroid rank sum valuations. Relevant concepts from matroid theory are reviewed in Appendix A.1.

Definition 5.2.1. *A set function $v : 2^{[m]} \to \mathbb{R}_+$ is a* matroid rank sum (MRS) *function if there exists a family of matroid rank functions $u_1, \ldots, u_\kappa : 2^{[m]} \to \mathbb{N}$, and associated non-negative weights $w_1, \ldots, w_\kappa \in \mathbb{R}^+$, such that $v(S) = \sum_{\ell=1}^{\kappa} w_\ell u_\ell(S)$ for all $S \subseteq [m]$.*

We do not assume any particular representation of MRS valuations, and require only oracle access to their (expected) values on certain distributions (see Section 5.2.4). MRS functions include most concrete examples of non-decreasing submodular functions that appear in the literature – this includes coverage functions,[8] matroid weighted-rank functions,[9] and all convex combinations thereof. Moreover, as shown in [54], $1-1/e$ is the best approximation possible in polynomial time for combinatorial auctions with MRS valuations unless $P = NP$, even ignoring strategic considerations. That being said, we note that some interesting submodular functions — such as some budget additive functions[10] — are not in the matroid rank sum family.

5.2.4 Lotteries and Oracles

Our result employs a randomized analogue of the value oracle model described in Section 5.2.2. Our oracle takes as input a description of a simple lottery over subsets

[8] A *coverage function* f on ground set $[m]$ designates some set $\mathcal{L}$ of elements, and m subsets $A_1, \ldots, A_m \subseteq \mathcal{L}$, such that $f(S) = |\cup_{j \in S} A_j|$. We note that $\mathcal{L}$ may be an infinite, yet measurable, space. Coverage functions are arguably *the* canonical example of a submodular function, particularly for combinatorial auctions.

[9] This is a generalization of matroid rank functions, where weights are placed on elements of the matroid. It is true, though not immediately obvious, that a matroid weighted-rank function can be expressed as a weighted combination of matroid (unweighted) rank functions – see e.g. [36].

[10] A set function f on ground set $[m]$ is *budgeted additive* if there exists a constant $B \geq 0$ (the budget) such that $f(S) = \min(B, \sum_{j \in S} f(\{j\}))$.

of $[m]$, and outputs the expectation of v over this lottery. Given a vector $x \in [0,1]^m$ of probabilities on the items, let D_x be the distribution over $S \subseteq [m]$ that includes each item j in S independently with probability x_j. We use $F_v(x)$ to denote the expected value of $v(S)$ over draws $S \sim D_x$ from this lottery.

Definition 5.2.2. *A* lottery-value oracle *for set function $v : 2^{[m]} \to \mathbb{R}$ takes as input a vector $x \in [0,1]^m$, and outputs*

$$F_v(x) = \mathop{\mathbf{E}}_{S \sim D_x}[v(S)] = \sum_{S \subseteq [m]} v(S) \prod_{j \in S} x_j \prod_{j \neq S} (1 - x_j). \tag{5.1}$$

We note that F_v is simply the well-studied *multi-linear extension* of v (see for example [19, 87]).

Value oracles are the traditional access model for a valuation function over bundles of items. We justify relaxing to lottery-value oracles on two grounds: First, a lottery-value oracle is easily implemented for various examples of MRS valuations, such as explicit coverage functions (see Section 5.6.1), making the two oracle models equivalent in these cases. Second, lottery-value oracles can be approximated arbitrarily well with high probability using a polynomial number of value oracle queries (see [87]). Even though we are not able to reconcile the incurred sampling errors — small as they may be — with the requirement that our mechanism be *exactly* truthful, we suspect that relaxing our solution concept to approximate truthfulness – also known as ϵ-truthfulness – would remove this difficulty, and allow us to relax our oracle model to the more traditional value oracles.

5.3 Result Statement and Proof Overview

We use the framework of Chapter 4 to design a mechanism for combinatorial auctions with MRS valuations in the lottery-value oracle model. Specifically we prove the following Theorem.

Theorem 5.3.1. *There is a* $(1-1/e)$*-approximate, truthful-in-expectation mechanism for combinatorial auctions with matroid rank sum valuations in the lottery-value oracle model, running in expected* $\mathrm{poly}(n, m)$ *time.*

We formulate combinatorial auctions as integer program (5.2). The feasible set Ω is encoded in $\{0,1\}^{n\times m}$, with variable x_{ij} indicating whether item j is allocated to player i. We use $w(x)$ to denote social welfare of the allocation described by variables x.

$$\begin{array}{lll} \text{maximize} & w(x) = \sum_i v_i(\{j : x_{ij} = 1\}) & \\ \text{subject to} & \sum_i x_{ij} \leq 1, & \text{for } j \in [m]. \\ & x_{ij} \in \{0,1\}, & \text{for } i \in [n], j \in [m]. \end{array} \tag{5.2}$$

We relax the feasible set of the above integer program as follows: Let $\mathcal{P} = \mathcal{P}(\Omega)$ be the result of relaxing the constraints $x_{ij} \in \{0,1\}$ of (5.2) to $0 \leq x_{ij} \leq 1$.

We define the allocation rule of our mechanism for combinatorial auctions as simply the instantiation of Algorithm 4.1 with a particular rounding scheme that we define in Section 5.4. We call our rounding scheme the *Poisson rounding scheme*, and we denote it by $r_{\text{poiss}} : \mathcal{P} \to dist(\Omega)$. As described in Chapter 4, implementing our allocation rule reduces to solving the following mathematical program.

$$\begin{array}{lll} \text{maximize} & f(x) = \mathbf{E}_{S\sim r_{\text{poiss}}(x)}[\sum_i v_i(S_i)] & \\ \text{subject to} & \sum_i x_{ij} \leq 1, & \text{for } j \in [m]. \\ & 0 \leq x_{ij} \leq 1, & \text{for } i \in [n], j \in [m]. \end{array} \tag{5.3}$$

Lemma 4.3.1 implies that our allocation rule is maximal in distributional range. Therefore, truth-telling payments can be generically computed — with polynomial overhead in runtime — as in Proposition 3.4.1. We prove in Section 5.4 that r_{poiss} is $(1 - 1/e)$-approximate (Lemma 5.4.2), and therefore by Lemma 4.3.2, so is our mechanism. More interestingly, we prove that r_{poiss} is convex (Lemma 5.4.1) in Section 5.5. Therefore, (5.3) is a convex program. Convex programs can frequently be solved in polynomial time, and as we show in Appendix B.1 ours is no exception. This completes the proof of Theorem 5.3.1.

5.4 The Poisson Rounding Scheme

We define the Poisson rounding scheme as follows. Given a fractional solution x to (5.3), do the following independently for each item j: assign j to player i with probability $1 - e^{-x_{ij}}$. (This is well defined since $1 - e^{-x_{ij}} \leq x_{ij}$ for all players i and items j, and $\sum_i x_{ij} \leq 1$ for all items j.) We make this precise in Algorithm 5.1. For clarity, we represent an allocation as a function from items to players, with an additional null player $*$ reserved for items that are left unassigned.

Input: Fractional allocation x with $\sum_i x_{ij} \leq 1$ for all j, and $0 \leq x_{ij} \leq 1$ for all i, j.
Output: Feasible allocation $a : [m] \to [n] \cup \{*\}$.

1: **for** $j = 1, \ldots, m$ **do**
2: Draw p_j uniformly at random from $[0, 1]$.
3: **if** $\sum_i (1 - e^{-x_{ij}}) \geq p_j$ **then**
4: Let $a(j)$ be the minimum index such that $\sum_{i \leq a(j)} (1 - e^{-x_{ij}}) \geq p_j$.
5: **else**
6: $a(j) = *$
7: **end if**
8: **end for**

Algorithm 5.1: The Poisson rounding scheme r_{poiss}.

The Poisson rounding scheme is $(1 - 1/e)$-approximate and convex. The proof of Lemma 5.4.2 is not difficult, and is included below. We prove Lemma 5.4.1 in Section 5.5.

Lemma 5.4.1. *The Poisson rounding scheme is convex when player valuations are matroid rank sum functions.*

Lemma 5.4.2. *The Poisson rounding scheme is $(1 - 1/e)$-approximate when player valuations are submodular.*

Proof. Let $S_1, \ldots, S_n$ be an allocation, and let x be an the integer point of (5.3) corresponding to $S_1, \ldots, S_n$. Let $(S'_1, \ldots, S'_n) \sim r_{\text{poiss}}(x)$. It suffices to show that $\mathbf{E}[\sum_i v_i(S'_i)] \geq (1 - 1/e) \cdot \sum_i v_i(S_i)$.

By definition of the Poisson rounding scheme, S_i' includes each $j \in S_i$ independently with probability $1 - 1/e$. Submodularity implies that $\mathbf{E}[v_i(S_i')] \geq (1 - 1/e) \cdot v_i(S_i)$ – this was proved in many contexts: see for example [42, Lemma 2.2], and the earlier related result in [41, Proposition 2.3]. This completes the proof. □

5.5 Convexity of the Poisson Rounding Scheme

First, we prove in Section 5.5.1 the special case of Lemma 5.4.1 for coverage valuations as a warmup. We then extend the proof to all MRS valuations in Section 5.5.2.

5.5.1 Warmup: Convexity for Coverage Valuations

Recall the definition of coverage valuations, given in Section 5.2.3. Fix n, m, and coverage valuations $\{v_i\}_{i=1}^n$, and let $\mathcal{P}$ denote the feasible set of mathematical program (5.3). Let $(S_1, \ldots, S_n) \sim r_{\text{poiss}}(x)$ be the (random) allocation computed by the Poisson rounding scheme for point $x \in \mathcal{P}$. The expected welfare $\mathbf{E}[w(r_{\text{poiss}}(x))]$ can be written as $\mathbf{E}[\sum_{i=1}^n v_i(S_i)]$, where the expectation is taken over the internal random coins of the rounding scheme. By linearity of expectation, as well as the fact that the sum of concave functions is concave, it suffices to show that $\mathbf{E}[v_i(S_i)]$ is a concave function of x for an arbitrary player i with coverage valuation v_i.

Fix player i, and use x_j, v, and S as short-hand for x_{ij}, v_i, and S_i respectively. Recall that v is a coverage function; let $\mathcal{L}$ be a ground set and $A_1, \ldots, A_m \subseteq \mathcal{L}$ be such that $v_i(T) = |\cup_{j \in T} A_j|$ for each $T \subseteq [m]$. The Poisson rounding scheme includes each item j in S independently with probability $1 - e^{-x_j}$. The expected value of player i can be written as follows.

$$\begin{aligned}\mathbf{E}[v(S)] &= \mathbf{E}[|\cup_{j \in S} A_j|] \\ &= \sum_{\ell \in \mathcal{L}} \mathbf{Pr}[\ell \in \cup_{j \in S} A_j]\end{aligned}$$

Since the sum of concave functions is concave, it suffices to show that $\mathbf{Pr}[\ell \in \cup_{j \in S} A_j]$ is concave in x for each $\ell \in \mathcal{L}$. We can interpret $\mathbf{Pr}[\ell \in \cup_{j \in S} A_j]$ as the probability

that element ℓ is *covered* by an item in S, where $j \in [m]$ covers $\ell \in \mathcal{L}$ if $\ell \in A_j$. For each $\ell \in \mathcal{L}$, let C_ℓ be the set of items that cover ℓ. Element $\ell \in \mathcal{L}$ is covered by S precisely when $C_\ell \cap S \neq \emptyset$. Each item $j \in C_\ell$ is included in S independently with probability $1 - e^{-x_j}$. Therefore, the probability $\ell \in \mathcal{L}$ is covered by S can be re-written as follows:

$$\begin{aligned}\mathbf{Pr}[\ell \in \cup_{j \in S} A_j] &= 1 - \prod_{j \in C_\ell} e^{-x_j} \\ &= 1 - \exp\left(-\sum_{j \in C_\ell} x_j\right). \end{aligned} \tag{5.4}$$

Form (5.4) is the composition of the concave function $g(y) = 1 - e^{-y}$ with the affine function $y \to \sum_{j \in C_\ell} x_j$. It is well-known that composing a concave function with an affine function yields another concave function (see e.g. [15]). Therefore, $\mathbf{Pr}[\ell \in \cup_{j \in S} A_j]$ is concave in x for each $\ell \in \mathcal{L}$, as needed. This completes the proof of Lemma 5.4.1 for the special case of coverage valuations.

5.5.2 Convexity for Matroid Rank Sum Valuations

We now prove Lemma 5.4.1 in its full generality. As a tool in our proof, we define a discrete analogue of a Hessian matrix for set functions, and show that these discrete Hessians are negative semi-definite for matroid rank sum functions.

Definition 5.5.1. *Let $v : 2^{[m]} \to \mathbb{R}$ be a set function. For $S \subseteq [m]$, we define the* discrete Hessian matrix $\mathcal{H}^v_S \in \mathbb{R}^{m \times m}$ *of v at S as follows:*

$$\mathcal{H}^v_S(j,k) = v(S \cup \{j,k\}) - v(S \cup \{j\}) - v(S \cup \{k\}) + v(S) \tag{5.5}$$

for $j, k \in [m]$.

Claim 5.5.2. *If $v : 2^{[m]} \to \mathbb{R}^+$ is a matroid rank sum function, then $\mathcal{H}^v_S$ is negative semi-definite for each $S \subseteq [m]$.*

Proof. We observe that $\mathcal{H}_S^v$ is linear in v, and recall that a non-negative weighted-sum of negative semi-definite matrices is negative semi-definite. Therefore, it is sufficient to prove this claim when v is a matroid rank function.

Let v be the matroid rank function of some matroid M with ground set $[m]$, and fix $S \subseteq [m]$. Observe that v is non-decreasing, submodular, integer-valued, and $v(T \cup \{j\}) \leq v(T) + 1$ for all $T \subseteq [m]$ and $j \in [m]$. Therefore, a simple case analysis reveals that for each $j, k \in [m]$

$$\mathcal{H}_S^v(j,k) = \begin{cases} -1 & \text{if } v(S \cup \{j\}) = v(S \cup \{k\}) = v(S \cup \{j,k\}) = v(S) + 1, \\ 0 & \text{otherwise.} \end{cases}$$

In other words, $-\mathcal{H}_S^v$ is a binary matrix where $-\mathcal{H}_S^v(j,k) = 1$ if and only if two conditions are satisfied: (1) Both $\{j\}$ and $\{k\}$ are independent sets in the contracted matroid M/S, and (2) $\{j,k\}$ is dependent in M/S.

It is clear that $-\mathcal{H}_S^v$ is symmetric. We now also show that $-\mathcal{H}_S^v$ encodes a *transitive* relation on $[m]$ — i.e. for all $j, k, \ell \in [m]$, if $-\mathcal{H}_S^v(j,k) = -\mathcal{H}_S^v(k,\ell) = 1$ then $-\mathcal{H}_S^v(j,\ell) = 1$. Fix j, k, ℓ such that $-\mathcal{H}_S^v(j,k) = -\mathcal{H}_S^v(k,\ell) = 1$. The sets $\{j\}$, $\{k\}$, and $\{\ell\}$ are independent sets of the contracted matroid M/S, and moreover $\{j,k\}$ and $\{k,\ell\}$ are dependent in M/S. Assume for a contradiction that $\{j,\ell\}$ is independent in M/S; applying the matroid exchange property to $\{k\}$ and $\{j,\ell\}$ implies that one of $\{j,k\}$ and $\{k,\ell\}$ must be independent in M/S as well, contradicting our choice of j, k, and ℓ. Therefore, $\{j,\ell\}$ is dependent in M/S, and $-\mathcal{H}_S^v(j,\ell) = 1$.

A binary matrix encoding a symmetric and transitive relation is a block diagonal matrix where each diagonal block is an all-ones or all-zeros sub-matrix. It is known, and easy to prove, that such a matrix is positive semi-definite. Therefore $\mathcal{H}_S^v$ is negative semi-definite. □

We now return to Lemma 5.4.1. Fix n, m, and MRS valuations $\{v_i\}_{i=1}^n$, and let $\mathcal{P}$ denote the feasible set of mathematical program (5.3). Let $(S_1, \ldots, S_n) \sim r_{\text{poiss}}(x)$ be the (random) allocation computed by the Poisson rounding scheme for point $x \in \mathcal{P}$. The expected welfare $\mathbf{E}[w(r_{\text{poiss}}(x))]$ can be written as $\mathbf{E}[\sum_{i=1}^n v_i(S_i)]$, where the

expectation is taken over the internal random coins of the rounding scheme. By linearity of expectation, as well as the fact that the sum of concave functions is concave, it suffices to show that $\mathbf{E}[v_i(S_i)]$ is a concave function of x for an arbitrary player i with MRS valuation v_i.

Fix player i, and use x_j, v, S as short-hand for x_{ij}, v_i, S_i respectively. The Poisson rounding scheme includes each item j in S independently with probability $1 - e^{-x_j}$. We can now write the expected value of player i as the following function $G_v : \mathbb{R}^m \to \mathbb{R}$:

$$G_v(x_1, \ldots, x_m) = \sum_{S \subseteq [m]} v(S) \prod_{j \in S} (1 - e^{-x_j}) \prod_{j \neq S} e^{-x_j} \tag{5.6}$$

The following claim, combined with Claim 5.5.2, completes the proof of Lemma 5.4.1.

Claim 5.5.3. *If all discrete Hessians of v are negative semi-definite, then G_v is concave.*

Proof. Assume $\mathcal{H}_S^v$ is negative semi-definite for each $S \subseteq [m]$. We work with G_v as expressed in Equation (5.6). We will show that the Hessian matrix of G_v at an arbitrary $x \in \mathbb{R}^m$ is negative semi-definite, which is a sufficient condition for concavity. We take the mixed-derivative of G_v with respect to x_j and x_k (possibly $j = k$).

$$\begin{aligned}
\frac{\partial^2 G_v(x)}{\partial x_j \partial x_k} &= \sum_{S \subseteq [m] \setminus \{j,k\}} \prod_{\ell \in S} (1 - e^{-x_\ell}) \prod_{\ell \in [m] \setminus S} e^{-x_\ell} \Big(v(S) - v(S \cup \{j\}) - v(S \cup \{k\}) \\
&\qquad\qquad + v(S \cup \{j,k\}) \Big) \\
&= \sum_{S \subseteq [m]} \prod_{\ell \in S} (1 - e^{-x_\ell}) \prod_{\ell \in [m] \setminus S} e^{-x_\ell} \Big(v(S) - v(S \cup \{j\}) - v(S \cup \{k\}) \\
&\qquad\qquad + v(S \cup \{j,k\}) \Big) \\
&= \sum_{S \subseteq [m]} \prod_{\ell \in S} (1 - e^{-x_\ell}) \prod_{\ell \in [m] \setminus S} e^{-x_\ell} \; \mathcal{H}_S^v(j,k)
\end{aligned}$$

The first equality follows by grouping the terms of Equation (5.6) by the projection of S onto $[m] \setminus \{i, j\}$, and then differentiating. The second equality follows from the

fact that $v(S) - v(S \cup \{j\}) - v(S \cup \{k\}) + v(S \cup \{j,k\}) = 0$ when S includes either of j and k. The last equality follows by definition of $\mathcal{H}^v_S$.

The above derivation immediately implies that we can write the Hessian matrix of $G_v(x)$ as a non-negative weighted sum of discrete Hessian matrices.

$$\bigtriangledown^2 G_v(x) = \sum_{S \subseteq [m]} \prod_{\ell \in S} (1 - e^{-x_\ell}) \prod_{\ell \in [m] \setminus S} e^{-x_\ell} \, \mathcal{H}^v_S \tag{5.7}$$

A non-negative weighted-sum of negative semi-definite matrices is negative semi-definite. This completes the proof of the claim. □

5.6 Additional Results

5.6.1 A Randomized Mechanism for Explicit Valuations

In this section, we apply our mechanism to explicitly represented coverage valuations. This demonstrates the utility of our mechanism in a concrete, non-oracle-based setting. Moreover, it stands in contrast to the impossibility result for deterministic VCG-based mechanisms in Section 5.6.2, and is a testament to the importance of randomization in the design of mechanisms for combinatorial auctions.

An n player, m item instance combinatorial auctions with *explicit coverage valuations* is described as follows. For each player i, there is a finite set $\mathcal{L}^i$, and a family $A^i_1, \ldots, A^i_m$ of subsets of $\mathcal{L}^i$. The valuation function of player i is then defined as $v_i(S) = |\cup_{j \in S} A^i_j|$. The set system $\left(\mathcal{L}^i, \left\{A^i_j\right\}_{j=1}^m\right)$ is encoded explicitly as a bipartite graph.

As discussed previously, MRS valuations include all coverage valuations. Therefore, in order to implement the mechanism of Theorem 5.3.1 for this problem, it suffices to answer lottery-value queries in time polynomial in the number of bits encoding the instance.

Claim 5.6.1. *In combinatorial auctions with explicit coverage valuations, lottery-value queries can be answered in time polynomial in the length of the encoding of the instance.*

Proof. Let $v : 2^{[m]} \to \mathbb{R}_+$ be a coverage valuation presented explicitly as a set system $(\mathcal{L}, \{A_j\}_{j=1}^m)$, and let $x \in [0,1]^m$. Let S be a random set that includes each $j \in [m]$ independently with probability x_j. The outcome of the lottery value oracle of v evaluated at x is equal to the sum, over all $\ell \in \mathcal{L}$, of the probability that ℓ is "covered" by S – specifically, $\sum_{\ell \in \mathcal{L}} \mathbf{Pr}[\ell \in \cup_{j \in S} A_j]$. It is easy to verify that a term of this sum can be expressed as the following closed form expression.

$$\mathbf{Pr}[\ell \in \cup_{j \in S} A_j] = 1 - \prod_{j : A_j \ni \ell} (1 - x_j)$$

This expression can be evaluated in time polynomial in the representation of the set system. This completes the proof. □

Claim 5.6.1 and Theorem 5.3.1 imply the following Theorem.

Theorem 5.6.2. *There is a $(1-1/e)$-approximate, truthful-in-expectation mechanism for combinatorial auctions with explicit coverage valuations that runs in polynomial time in expectation.*

5.6.2 An Impossibility for VCG-based Deterministic Mechanisms

In this section, we show that deterministic techniques for the design of VCG-based mechanisms fall short of providing a constant approximation factor for combinatorial auctions with explicit coverage valuations. This stands in contrast with the result in Section 5.6.1 for randomized mechanisms.

We use the following special case of [17, Theorem 1.2]: If a succinct combinatorial auction problem satisfies the *regularity* conditions on the valuations defined in [17], and moreover the 2-player version of the problem is APX hard, then no polynomial-time maximal-in-range[11] algorithm guarantees an approximation ratio of $o(n)$.

It is routine to verify the regularity assumptions of [17] for explicit coverage valuations. APX-hardness of the 2-player problem follows by an elementary reduction from

[11] Recall the definition of maximal-in-range algorithms from Section 3.3.

the APX-hard problem *max-cut*. Given an instance of max-cut on a graph $G = (V, E)$, we let $[m] = V$, $\mathcal{L}^1 = \mathcal{L}^2 = E$. For $e \in E$, $i \in \{1, 2\}$, and $j \in V$, we let $e \in A^i_j$ if j is one of the endpoints of edge e. It is easy to check that the welfare-maximizing allocation of the resulting 2-player instance of combinatorial auctions corresponds to the maximum cut of G. Moreover, using the fact that the optimal objective value of max-cut is at least $|E|/2$, it is elementary to verify that the reduction preserves hardness of approximation up to a constant factor. Therefore, combinatorial auctions with explicit coverage valuations and 2 players is APX hard. This yields the following Theorem.

Theorem 5.6.3. *No polynomial-time maximal-in-range algorithm for combinatorial auctions with explicit coverage valuations achieves a approximation ratio of $o(n)$, unless $NP \subseteq P/poly$.*

Chapter 6

Combinatorial Public Projects

6.1 Introduction

In combinatorial public projects, a public planner must choose from a set of projects to undertake subject to a resource constraint, and a set of self-interested players have private valuations over packages of projects. Combinatorial public projects has emerged as the paradigmatic "hard problem" of algorithmic mechanism design, due to a large body of impossibility results for variants of the problem.

We use the convex rounding framework introduced in Chapter 4 to design an expected-polynomial-time, truthful-in-expectation, $(1 - 1/e)$-approximation mechanism for welfare maximization in a fundamental variant of combinatorial public projects. Our approximation factor is the best possible for the variant of combinatorial public projects we consider, assuming $P \neq NP$. Our result is the first truthful-in-expectation and polynomial-time mechanism to achieve a constant-factor approximation for a natural NP-hard variant of combinatorial public projects, and stands in contrast to the body of existing negative results for this problem.

6.1.1 Summary of Results and Techniques

In *combinatorial public projects*, there is a set of projects that may be undertaken, and a set of self-interested players each equipped with a private valuation over subsets

of the projects. The principal is a public planner who must choose a subset of these projects, subject to choosing at most a given number k of the projects, with the goal of maximizing social welfare. Approximation mechanisms for combinatorial public projects, in particular the variant with submodular valuations,[1] have received much attention in algorithmic mechanism design in recent years, mostly in the form of negative results.

Using the convex rounding framework introduced in Chapter 4, we design an expected-polynomial-time, truthful-in-expectation, $(1 - 1/e)$-approximation mechanism for welfare maximization in combinatorial public projects, when players have valuations that are *matroid rank sums (MRS)*. This is the same class of valuations considered for combinatorial auctions in Chapter 5, and we recall that it includes most concrete examples of submodular functions studied in a resource allocation context, such as coverage functions, matroid weighted-rank functions, and convex combinations thereof. Our approximation factor is the best possible, even for known and explicitly given coverage valuations, assuming $P \neq NP$.

To prove our result, we follow the general outline of Chapter 5. However, our task is more challenging: whereas in combinatorial auctions, randomized rounding may allocate each item independently (the approach taken in Chapter 5), this is not possible for CPP. We must respect the cardinality constraint of k on the set of chosen projects, and therefore our rounding scheme must by fiat be *dependent*. This presents a major challenge in analyzing our rounding scheme. Whereas the expected value of a submodular function on a product distribution (i.e. independent rounding) has been studied extensively, and is closely related to the now well-understood multi-linear extension (see e.g. [19, 87]), analyzing the expected value of a dependent distribution — in particular proving it to be a concave function of underlying parameters — is a technical challenge that we overcome by combining techniques from combinatorics, convex analysis, and matroid theory.

[1] As mentioned in Chapter 5, submodular functions are those set functions v that satisfy *diminishing marginal returns*: specifically, the marginal value $v(S \cup \{j\}) - v_i(S)$ for a each fixed item j is non-increasing in S.

6.1.2 Related Work

As mentioned previously, most prior results in algorithmic mechanism design for combinatorial public projects are negative, and none have exhibited constant-factor approximation mechanisms for a natural NP-hard variant of the problem. We overview the negative results in related work next.

Combinatorial public projects, in particular its *exact* variant,[2] was first introduced by Papadimitriou et al. [74]. They show that no deterministic truthful mechanism for exact CPP with submodular valuations can guarantee better than a $1/O(\sqrt{m})$ fraction of the optimal social welfare. This stands in contrast to the non-strategic version of the problem, which admits a $(1 - 1/e)$-approximation algorithm due to Nemhauser, Wolsey and Fisher [69], and this is optimal [75] assuming $P \neq NP$.

Buchfuhrer et al. [18] explored approximation algorithms and truthful mechanisms for CPP with various classes of valuations in the submodular hierarchy. The most relevant result of [18] to our work is a lower bound of $\Omega(\sqrt{m})$ on the approximation ratio of *deterministic* truthful mechanisms for the exact variant of CPP with coverage valuations — a class of valuations for which our *randomized* mechanism for flexible CPP obtains a $(1 - 1/e)$ approximation.

Recently, Dobzinski [26] showed two lower bounds for CPP in the value oracle model: A lower bound of $\Omega(\sqrt{m})$ on universally truthful mechanisms for flexible CPP with submodular valuations, and a lower bound of $\Omega(\sqrt{m})$ on truthful-in-expectation mechanisms for *exact* CPP with submodular valuations. We note that the latter was the first unconditional lower bound on truthful-in-expectation mechanisms. Most recently, Dughmi and Vondrák showed that no truthful-in-expectation and polynomial-time mechanism for submodular combinatorial public projects in the value oracle model provides an approximation guarantee better than $m^{-\alpha}$, for some specific constant α. Together, these results imply that extending the results of this chapter to submodular valuations seems unlikely without additional assumptions.

[2]The exact variant of combinatorial public projects constrains feasible sets to contain exactly k projects, rather than at most k as in the flexible variant we consider here.

6.2 Model and Preliminaries

6.2.1 Combinatorial Public Projects

In *combinatorial public projects (CPP)* there is a set $[m] = \{1, \ldots, m\}$ of *projects*, a cardinality bound k such that $0 \leq k \leq m$, and a set $[n] = \{1, \ldots, n\}$ of *players*. Each player i has a valuation function $v_i : 2^{[m]} \to \mathbb{R}_+$ that is normalized ($v_i(\emptyset) = 0$) and non-decreasing ($v_i(A) \leq v_i(B)$ whenever $A \subseteq B$). In this chapter, we consider the *flexible* variant of combinatorial public projects: a feasible solution is a set $S \subseteq [m]$ of projects with $|S| \leq k$. Player i's value for outcome S is equal to $v_i(S)$. We consider CPP when the goal is to choose the feasible set S maximizing *social welfare*: $\sum_i v_i(S)$.

Our result will assume that each player's valuation function is a *matroid rank sum (MRS)* function (Section 5.2.3), presented via a *bounded-lottery-value oracle* (Section 6.2.2).

6.2.2 Lotteries and Oracles

As mentioned in Section 5.2.2, it is traditional to model access to a valuation as a *value oracle*. For a valuation $v : 2^{[m]} \to \mathbb{R}$, this oracle takes as input a set $S \subseteq [m]$, and returns $v(S)$. Similar to Chapter 5, this chapter uses a randomized analogue of value oracles; one that takes in a description of a simple lottery over sets $S \subseteq [m]$, and outputs the expectation of v over this lottery.

Let $k \in [m]$, let $R \subseteq [m]$, and let $x \in [0, 1]^m$ be a vector such that $\sum_j x_j \leq 1$. We interpret x as a probability distribution over $[m] \cup \{*\}$, where $*$ represents not choosing a project. Specifically, project $j \in [m]$ is chosen with probability x_j, and $*$ is chosen with probability $1 - \sum_j x_j$. We define a distribution $D_k^R(x)$ over $2^{[m]}$, and call this distribution the *k-bounded lottery with marginals x and promise R*. We sample $S \sim D_k^R(x)$ as follows: Let $j_1, \ldots, j_k$ be independent draws from x, and let $S = R \cup \{j_1, \ldots, j_k\} \setminus \{*\}$. Essentially, this lottery commits to choosing projects R, and adds an additional k projects chosen randomly with replacement from distribution x. When $R = \emptyset$, as will be the case through most of this paper, we omit mention of the promised set. Our oracles return the expected value of a bounded lottery.

Definition 6.2.1. *A* bounded-lottery-value oracle *for set function* $v : 2^{[m]} \to \mathbb{R}$ *takes as input a vector* $x \in [0,1]^m$ *with* $\sum_j x_j \leq 1$*, a bound* $k \in [m]$*, and a set* $R \subseteq [m]$*, and outputs* $\mathbf{E}_{S \sim D_k^R(x)}[v(S)]$*.*

In our model for CPP, we assume that a player with valuation function v_i can answer bounded-lottery-value oracle queries for v_i. A bounded-lottery-value oracle is a generalization of the more traditional value oracles. Nevertheless, as we did in Chapter 5, we justify our stronger oracle model on two grounds: First, a bounded-lottery-value oracle is easily implemented for various examples of MRS valuations, such as explicit coverage functions (not too dissimilar from the implementation of Section 5.6.1), making the two oracle models equivalent in these cases. Second, bounded-lottery-value oracles can be approximated arbitrarily well with high probability using a polynomial number of value oracle queries; this is done by random sampling, and we omit the straightforward analysis. Even though we are not able to reconcile the incurred sampling errors — small as they may be — with the requirement that our mechanism be *exactly* truthful, we suspect that relaxing our solution concept to approximate truthfulness – also known as ϵ-truthfulness – would remove this difficulty, and allow us to relax our oracle model to the more traditional value oracles.

6.3 Result Statement and Proof Overview

We use the framework of Chapter 4 to design a mechanism for combinatorial public projects with MRS valuations in the bounded-lottery-value oracle model. Specifically we prove the following Theorem.

Theorem 6.3.1. *There is a* $(1-1/e)$*-approximate, truthful-in-expectation mechanism for combinatorial public projects with matroid rank sum valuations in the bounded-lottery-value oracle model, running in expected* $\text{poly}(n, m)$ *time.*

We encode combinatorial public projects as integer program (6.1). The feasible set Ω is encoded in $\{0,1\}^m$, with variable x_j indicating whether project j is chosen.

We use $w(x)$ to denote social welfare of the set of projects described by variables x.

$$\begin{array}{lll} \text{maximize} & w(x) = \sum_i v_i(\{j : x_j = 1\}) & \\ \text{subject to} & \sum_{j=1}^{m} x_j \leq k & \\ & x_j \in \{0,1\}, & \text{for } j \in [m]. \end{array} \tag{6.1}$$

We relax the feasible set of the above integer program as follows: Let $\mathcal{P} = \mathcal{P}(\Omega)$ be the result of relaxing the constraints $x_j \in \{0,1\}$ of (6.1) to $0 \leq x_j \leq 1$.

We define the allocation rule of our mechanism for combinatorial public projects as simply the instantiation of Algorithm 4.1 with a particular rounding scheme that we define in Section 6.4. We call our rounding scheme the *k-bounded-lottery rounding scheme*, and we denote it by $r_k : \mathcal{P} \to dist(\Omega)$. As described in Chapter 4, implementing our allocation rule reduces to solving the following mathematical program.

$$\begin{array}{lll} \text{maximize} & f(x) = \mathbf{E}_{S \sim r_k(x)}[\sum_i v_i(S)] & \\ \text{subject to} & \sum_{j=1}^{m} x_j \leq k & \\ & 0 \leq x_j \leq 1, & \text{for } j \in [m]. \end{array} \tag{6.2}$$

The outline of the proof in this chapter is essentially identical to that of Chapter 5. Lemma 4.3.1 implies that our allocation rule is maximal in distributional range. Therefore, truth-telling payments can be generically computed — with polynomial overhead in runtime — as in Proposition 3.4.1. We prove in Section 6.4 that r_k is $(1-1/e)$-approximate (Lemma 6.4.2), and therefore by Lemma 4.3.2, so is our mechanism. More interestingly, we prove that r_k is convex (Lemma 6.4.1) in Section 6.5. Therefore (6.2) is a convex program, and we show that it can be solved in polynomial time in Appendix B.2. This completes the proof of Theorem 6.3.1.

6.4 The k-Bounded-Lottery Rounding Scheme

We define the *k-bounded-lottery rounding scheme* r_k as follows. Given a vector x feasible for (6.2), we let distribution $r_k(x)$ be the k-bounded-lottery with marginals x/k (and promise $\emptyset$), as defined in Section 6.2.2. We make this precise in Algorithm 6.1.

Input: Fractional solution $x \in \mathbb{R}^m$ with $\sum_j x_j \leq k$, and $0 \leq x_j \leq 1$ for all j.
Output: $S \subseteq [m]$ with $|S| \leq k$
1: For each $j \in [m]$ designate the interval $I_j = [\frac{1}{k}\sum_{j'<j} x_{j'}, \frac{1}{k}\sum_{j'\leq j} x_{j'}]$ of length $\frac{x_j}{k}$
2: Draw $p_1, \ldots, p_k$ independently and uniformly from $[0, 1]$
3: Let $S = \{j \in [m] : \{p_1, \ldots, p_k\} \cap I_j \neq \emptyset\}$

Algorithm 6.1: The k-bounded-lottery rounding scheme r_k.

The k-bounded-lottery rounding scheme is $(1-1/e)$-approximate and convex. We prove the approximation lemma below. The convexity lemma is proved in Section 6.5.

Lemma 6.4.1. *The k-bounded-lottery rounding scheme is convex for CPP with MRS valuations.*

Lemma 6.4.2. *The k-bounded-lottery rounding scheme is $(1-1/e)$-approximate when valuations are submodular.*

Proof. Fix n, m, k and $\{v_i\}_{i=1}^n$. Let $S \subseteq [m]$ be a feasible solution to CPP — i.e. $|S| \leq k$. Let 1_S be the vector with 1 in indices corresponding to S, and 0 otherwise. Let $T \sim r_k(1_S)$. We will first show that each element of $j \in S$ is included in T with probability at least $1 - 1/e$. Observe that T is the union of k independent draws from a distribution on $[m] \cup \{*\}$, where each time the probability of $j \in S$ is $1/k$. Therefore, the probability that j is included in T is $1 - (1 - 1/k)^k \geq 1 - 1/e$.

Submodularity now implies that $\mathbf{E}[v_i(T)] \geq (1 - 1/e) \cdot v_i(S)$ for each player i — this was proved in many contexts: see for example [42, Lemma 2.2], and the earlier related result in [41, Proposition 2.3]. This completes the proof. □

6.5 Convexity of the k-Bounded-Lottery Rounding Scheme

First, we prove in Section 6.5.1 the special case of Lemma 6.4.1 for coverage valuations as a warmup. We then extend the proof to all MRS valuations in Section 6.5.2.

6.5.1 Warmup: Convexity for Coverage Valuations

As defined in Chapter 5, a coverage function f on ground set $[m]$ designates some set $\mathcal{Y}$, and m subsets $A_1, \ldots, A_m \subseteq \mathcal{Y}$, such that $f(S) = |\cup_{j \in S} A_j|$.

Fix n, m, k and $\{v_i\}_{i=1}^n$. Assume that, for each player i, the valuation function $v_i : 2^{[m]} \to \mathbb{R}$ is a coverage function. We let $v(S) = \sum_i v_i(S)$ be the welfare of a solution S to CPP. It is an easy observation that the sum of coverage functions is also a coverage function. Therefore $v(S)$ is a coverage function. We let $\mathcal{Y}$ be a set, and $A_1, \ldots, A_m \subseteq \mathcal{Y}$, such that $v(S) = |\cup_{j \in S} A_j|$. While our proof extends easily to the case where $\mathcal{Y}$ is an arbitrary measure space, we assume in this section that $\mathcal{Y}$ is a finite set for simplicity.

Let $\mathcal{P}$ denote the polytope of fractional solutions to CPP as given in (6.2). We now show that $\mathbf{E}_{S \sim r_k(x)}[v(S)]$ is a concave function of x for $x \in \mathcal{P}$, completing the proof of Lemma 6.4.1 for the special case of coverage valuations. Take an arbitrary $x \in \mathcal{P}$, and let $S \sim r_k(x)$ be a random variable. Using linearity of expectations, we can rewrite the expected welfare as follows.

$$\mathbf{E}[v(S)] = \mathbf{E}[|\cup_{j \in S} A_j|] = \sum_{\ell \in \mathcal{Y}} \mathbf{Pr}[\ell \in \cup_{j \in S} A_j]$$

Since the sum of concave functions is concave, showing that $\mathbf{Pr}[\ell \in \cup_{j \in S} A_j]$ is concave in x for each $\ell \in \mathcal{Y}$ suffices to complete the proof. For $\ell \in \mathcal{Y}$, let $T_\ell = \{j \in [m] : \ell \in A_j\}$ be the set of projects that "cover" ℓ. Let $p_1, \ldots, p_k$ and $I_1, \ldots, I_k$ be as in Algorithm 6.1. Note that $\{I_j\}_{j=1}^m$ are disjoint sub-intervals of $[0, 1]$, and $|I_j| = \frac{x_j}{k}$. We can rewrite the probability of covering ℓ as follows.

$$\begin{aligned}
\mathbf{Pr}[\ell \in \cup_{j\in S} A_j] &= \mathbf{Pr}[S \cap T_\ell \neq \emptyset] \\
&= \mathbf{Pr}[\{p_1, \ldots, p_k\} \cap \cup_{j\in T_\ell} I_j \neq \emptyset] \\
&= 1 - \mathbf{Pr}[\{p_1, \ldots, p_k\} \cap \cup_{j\in T_\ell} I_j = \emptyset] \\
&= 1 - \prod_{t=1}^{k} \mathbf{Pr}[p_t \notin \cup_{j\in T_\ell} I_j] \\
&= 1 - \prod_{t=1}^{k} (1 - |\cup_{j\in T_\ell} I_j|) \\
&= 1 - \left(1 - \frac{\sum_{j\in T_\ell} x_j}{k}\right)^k.
\end{aligned}$$

The final form is simply the composition of the concave function $g(y) = 1-(1-y/k)^k$ with the affine function $y \to \sum_{j\in T_\ell} x_j$. It is well known that composing a concave function with an affine function yields another concave function (see e.g. [15]). This completes the proof of Lemma 6.4.1 for the special case of coverage valuations.

6.5.2 Convexity for Matroid Rank Sum Valuations

We now prove Lemma 6.4.1 in its full generality. First, we recall the *discrete hessian matrix*, as defined in Chapter 5.

Definition 6.5.1 (Chapter 5)**.** *Let $v : 2^{[m]} \to \mathbb{R}$ be a set function. For $S \subseteq [m]$, we define the discrete Hessian matrix $\mathcal{H}^v_S \in \mathbb{R}^{m\times m}$ of v at S as follows:*

$$\mathcal{H}^v_S(i,j) = v(S \cup \{i,j\}) - v(S \cup \{i\}) - v(S \cup \{j\}) + v(S) \tag{6.3}$$

for $i, j \in [m]$.

It was shown in Chapter 5 that the discrete hessian matrices are negative semi-definite for matroid rank sum functions.

Claim 6.5.2 (Chapter 5)**.** *If $v : 2^{[m]} \to \mathbb{R}^+$ is a matroid rank sum function, then $\mathcal{H}^v_S$ is negative semi-definite for each $S \subseteq [m]$.*

We now return to Lemma 6.4.1. Fix n and m. For each cardinality bound $k \in [m]$, let $\mathcal{P}_k$ denote the polytope of fractional solutions to CPP as given in (6.2). For a set of MRS valuations $v_1, \ldots, v_n$, we observe that the social welfare $v(S) = \sum_{i=1}^{n} v_i(S)$ is — by the (obvious) fact that the sum of MRS functions is an MRS function — also an MRS function. Therefore, we will prove Lemma 6.4.1 by showing that, for each $k \in [m]$ and MRS function $v : 2^{[m]} \to \mathbb{R}$, the following function of $x \in \mathcal{P}_k$ is concave in x.

$$\begin{aligned} G_k^v(x) &= \mathop{\mathbf{E}}_{S \sim r_k(x)}[v(S)] \\ &= \sum_{S \subseteq [m]} v(S) \, \mathbf{Pr}[r_k(x) = S] \end{aligned} \tag{6.4}$$

We use techniques from combinatorics to write $\mathbf{Pr}[r_k(x) = S]$ in a form that will be easier to work with. For $T \subseteq [m]$, we use x_T as short-hand for $\sum_{j \in T} x_j$, and $\overline{T}$ as short-hand for $[m] \setminus T$.

Claim 6.5.3. *For each $k \in [m]$, $x \in \mathcal{P}_k$, and $S \subseteq [m]$*

$$\mathbf{Pr}[r_k(x) = S] = -1^{|S|} \sum_{R \subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^k \tag{6.5}$$

Proof. It is easy to see that $\mathbf{Pr}[r_k(x) = S]$ is equal to:

$$\mathbf{Pr}[r_k(x) \subseteq S] - \mathbf{Pr}[\bigvee_{j \in S} r_k(x) \subseteq S \setminus \{j\}] \tag{6.6}$$

Using the inclusion-exclusion principle, we can rewrite (6.6) as follows:

$$\mathbf{Pr}[r_k(x) \subseteq S] - \sum_{\emptyset \neq T \subseteq S} -1^{|T|-1} \, \mathbf{Pr}[r_k(x) \subseteq S \setminus T] \tag{6.7}$$

Letting $R = S \setminus T$ in (6.7), we get

$$\mathbf{Pr}[r_k(x) \subseteq S] - \sum_{R \subsetneq S} -1^{|S|-|R|-1} \, \mathbf{Pr}[r_k(x) \subseteq R] \tag{6.8}$$

We can easily simplify (6.8) to conclude that

$$\mathbf{Pr}[r_k(x) = S] = \sum_{R \subseteq S} -1^{|S|-|R|} \mathbf{Pr}[r_k(x) \subseteq R] \tag{6.9}$$

Next, we observe that the expression $\mathbf{Pr}[r_k(x) \subseteq R]$ can be expressed as a simple closed form in x. Let $p_1, \ldots, p_k$ and $I_1, \ldots, I_m$ be as in Algorithm 6.1. The event $r_k(x) \subseteq R$ occurs exactly when none of $p_1, \ldots, p_k$ land in the intervals corresponding to projects $\overline{R}$. Recalling that the interval I_j of project j has length x_j/k, we get that the probability of any particular p_t falling in $\cup_{j \in \overline{R}} I_j$ is exactly $x_{\overline{R}}/k$. Therefore, by the independence of the variables $p_1, \ldots, p_k$, we get that

$$\mathbf{Pr}[r_k(x) \subseteq R] = \left(1 - \frac{x_{\overline{R}}}{k}\right)^k \tag{6.10}$$

Combining (6.9) and (6.10) completes the proof. □

Building on Claim 6.5.3, we now express the Hessian matrix of G_k^v as a non-negative weighted sum of discrete Hessian matrices of v. We note that when $x \in \mathcal{P}_k$, it is easy to verify that $\frac{k-2}{k} \cdot x \in \mathcal{P}_{k-2}$, and therefore (6.11) is well-defined.

Claim 6.5.4. *For each $k \in [m]$, $x \in \mathcal{P}_k$, and $v : 2^{[m]} \to \mathbb{R}$, we have*

$$\bigtriangledown^2 G_k^v(x) = \frac{k-1}{k} \sum_{S \subseteq [m]} \mathbf{Pr}\left[r_{k-2}\left(\frac{k-2}{k} \cdot x\right) = S\right] \mathcal{H}_S^v \tag{6.11}$$

Proof. Fix $i, j \in [m]$, possibly with $i = j$. We work with G_k^v as defined in Equation (6.4), and plug in expression (6.5).

$$G_k^v(x) = \sum_{S \subseteq [m]} v(S) \cdot -1^{|S|} \sum_{R \subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^k$$

Differentiating with respect to x_i and x_j gives:

$$\frac{\partial^2 G_k^v(x)}{\partial x_i \partial x_j} = \frac{k-1}{k} \sum_{S \subseteq [m]} v(S) \cdot -1^{|S|} \sum_{R \subseteq S \setminus \{i,j\}} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-2}$$

We group the terms by projecting S onto $[m]\backslash\{i,j\}$, and then we simplify the resulting expression.

$$\begin{aligned}\frac{\partial^2 G_k^v(x)}{\partial x_i \partial x_j} =& \frac{k-1}{k} \sum_{S\subseteq[m]\backslash\{i,j\}} -1^{|S|} \sum_{R\subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-2} \Big(v(S) - v(S\cup\{i\}) \\ & - v(S\cup\{j\}) + v(S\cup\{i,j\})\Big) \\ =& \frac{k-1}{k} \sum_{S\subseteq[m]} -1^{|S|} \sum_{R\subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-2} \Big(v(S) - v(S\cup\{i\}) - v(S\cup\{j\}) \\ & + v(S\cup\{i,j\})\Big) \\ =& \frac{k-1}{k} \sum_{S\subseteq[m]} -1^{|S|} \sum_{R\subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-2} \mathcal{H}_S^v(i,j) \qquad (6.12)\end{aligned}$$

The second equality follows from the fact that $v(S) - v(S\cup\{i\}) - v(S\cup\{j\}) + v(S\cup\{i,j\}) = 0$ when S includes either of i and j. The last equality follows by definition of $\mathcal{H}_S^v$.

Invoking Claim 6.5.3 with $k' = k-2$ and $x' = \frac{k-2}{k}\cdot x$, and plugging the resulting expression into into (6.12), we conclude that

$$\frac{\partial^2 G_k^v(x)}{\partial x_i \partial x_j} = \frac{k-1}{k} \sum_{S\subseteq[m]} \mathbf{Pr}\left[r_{k-2}\left(\frac{k-2}{k}\cdot x\right) = S\right] \mathcal{H}_S^v(i,j).$$

□

Claims (6.5.2) and (6.5.4) establish that, when v is MRS and $k \in [m]$, $\bigtriangledown^2 G_k^v(x)$ is a non-negative weighted sum of negative semi-definite matrices for each $x \in \mathcal{P}_k$. A non-negative weighted sum of negative semi-definite matrices is negative semi-definite. Therefore, the Hessian matrix of G_k^v is negative semi-definite at each $x \in \mathcal{P}_k$, and we conclude that G_k^v is a concave function on $\mathcal{P}_k$. This completes the proof of Lemma 6.4.1.

Part III

Perturbation-Based Mechanisms

Chapter 7

The Linear Perturbation Framework

7.1 Introduction

In this chapter, we introduce a new framework for designing approximation mechanisms based on perturbing instances of a welfare-maximization mechanism design problem. Our approach is motivated by work from the celebrated field of smoothed analysis of algorithms, which shows that optimization problems with a perturbed objective function are sometimes more tractable than their worst-case counterparts. In particular, we consider algorithms for mechanism design problems that randomly perturb the social welfare objective function, and then find the allocation maximizing the perturbed welfare. Under some conditions, these algorithms can be implemented in expected polynomial time, and result in an approximately welfare-maximizing allocation.

These perturbation-based algorithms can not directly be used as part of a truthful-in-expectation mechanism, motivating the main idea behind our approach. We introduce a *duality* between perturbing an objective function and perturbing a feasible set. Specifically, we show that maximizing the perturbed welfare can be reinterpreted as maximizing the un-perturbed welfare over a set of *perturbed allocations*. When

certain conditions are satisfied, the perturbed allocations can be interpreted as a distributional range, yielding a maximal-in-distributional-range allocation rule that is implementable as part of a truthful-in-expectation mechanism.

7.1.1 Summary of Results and Techniques

Our framework applies to welfare maximization mechanism design problems that are *linear*. A feasible set Ω for such a problem is encoded in some euclidean space $\mathbb{R}^d$, and each player i's valuation function u_i is linear in this encoding — specifically, $u_i(x) = \sum_{j=1}^{d} u_{ij}x_j$ for all $x \in \Omega$. When $v = \sum_i u_i$, the social welfare of outcome x is equal to $v^T x$. One example of this class of problems is the knapsack allocation problem introduced in Chapter 1. (See Section 8.5 for several additional examples.)

We consider problems in this class which, like the knapsack allocation problem, are intractable in the worst case, but can be solved efficiently if a small perturbation is applied to their linear objective function v (social welfare in our setting). Specifically, a perturbation of the objective function is a (random) map $v \to \widehat{v}$, typically chosen so $\widehat{v}$ is "close" to v. Results from smoothed complexity (Section 8.2.2) suggest that many problems that are NP-hard in the worst case can be solved in expected-polynomial-time over random choices of $\widehat{v}$, provided that $\widehat{v}$ is "sufficiently random".

As a naive starting point, suppose we apply a perturbation to a given instance (Ω, v) of a linear welfare-maximization problem Π, and then invoke a polynomial-time algorithm to compute an optimal solution to the perturbed instance $(\Omega, \widehat{v})$. The good news is that this outcome will be near-optimal for the unperturbed instance provided that $\widehat{v}$ is "close" to v. The bad news is that exact optimization using perturbed valuations is not generally implementable as part of a truthful mechanism. Even when only a single player is present, strategic misreporting of his valuation v may yield a perturbed report that is preferred over $\widehat{v}$; i.e. one that leads to a better outcome with respect to his true valuation v. Are there perturbations that can be used for the design of truthful mechanisms, and moreover are expressive enough to reduce the complexity of interesting optimization problems?

We advocate the design of *linear* perturbations for this purpose, and we develop a "duality theory" to facilitate their application to the design of truthful mechanisms. A *linear perturbation* is given by a distribution over $d \times d$ matrices. Given objective function v and a (random) matrix P drawn from the perturbation, the perturbed objective function is defined as Pv. We observe that exact maximization of the perturbed objective function Pv over the feasible outcomes Ω of an instance is equivalent to exact maximization of the true objective function v over a set of "perturbed outcomes" $P^T\Omega$ with the "adjoint" perturbation matrix P^T. When P satisfies certain conditions, each such perturbed outcome in $P^T\Omega$ can be expressed as a probability distribution over outcomes in Ω. In this case, the "adjoint problem" can be solved truthfully via a maximal-in-distributional-range algorithm. Thus the "dual perspective" and the use of perturbed outcomes allow us to argue truthfulness for perturbations that seem, at first blush, fundamentally incompatible with truthful mechanisms.

7.2 Model and Preliminaries

7.2.1 Linear Optimization Problems

An optimization problem Π is *linear* if, for each instance $(\Omega, v) \in \Pi$, outcomes $x \in \Omega$ are encoded in some euclidean space $\mathbb{R}^d$, and v is a linear function of this encoding. For these problems, we naturally interpret v as a vector in $\mathbb{R}^d$ as well. We consider *non-negative linear maximization problems* in this chapter, where $\Omega \subseteq \mathbb{R}^d_+$, $v \in \mathbb{R}^d_+$, and the goal is to find $x \in \Omega$ maximizing $v^T x$.

We are mainly interested in problems where the objective function $v^T x$ is the welfare of self-interested players with private valuation functions. Consider a feasible set $\Omega \subseteq \mathbb{R}^d_+$ and n players, where player i has valuation $\sum_{j=1}^{d} u_{ij} x_j$ for each $x \in \Omega$. The corresponding welfare maximization problem — computing the outcome x that maximizes the sum of players' values — is then the linear maximization problem with $v_j = \sum_{i=1}^{n} u_{ij}$ for each $j = 1, 2, \ldots, d$. We next give a simple example to make these definitions concrete for the reader; see Section 8.5 for several more.

Example 7.2.1 (Knapsack Allocation)**.** *In the* knapsack allocation *problem, there are m projects and n players. Each project j has a publicly known cost s_j, and the feasible sets correspond to subsets of projects that have total cost at most a publicly known budget C. Each player i has a private value u_{ij} for each project j. Welfare maximization for knapsack allocation is a linear maximization problem: the feasible set is naturally encoded as the vectors x in $\{0,1\}^m$ with $\sum_j s_j x_j \leq C$, and the coefficient v_j is defined as the total value $\sum_i u_{ij}$ to all players of selecting project j.*

7.2.2 Perturbations

A *perturbation* for an optimization problem Π is a randomized algorithm Ψ that takes as input an instance (Ω, v) of Π and outputs another objective function $\widehat{v} = \Psi(\Omega, v)$. When Π is a non-negative maximization problem and $\epsilon > 0$, we say such a perturbation is *ϵ-approximation preserving* if for every instance (Ω, v) of Π, and outcome $x \in \Omega$, the absolute difference between the (expected) perturbed objective value of x and its unperturbed objective value is at most an ϵ fraction of the value of the optimum solution; formally, $|\mathbf{E}[\widehat{v}^T x] - v^T x| \leq \epsilon \max_{y \in \Omega} v^T y$, where the expectation is over the random coin flips of the perturbation.

7.3 An Overly Simplistic Approach

Suppose we design an exact algorithm $\mathcal{A}$ and an ϵ-approximation preserving perturbation Ψ for a non-negative maximization problem Π such that, for every instance (Ω, v), algorithm $\mathcal{A}$ has expected running time polynomial in the instance size when the instance is perturbed by Ψ. Since Ψ is ϵ-approximation preserving, we immediately get a $(1 - 2\epsilon)$-approximation algorithm for Π: given an instance of Π, use Ψ to perturb the instance and the algorithm $\mathcal{A}$ to efficiently solve the perturbed instance.

Can we design such a perturbation so that the resulting approximation algorithm can be used in a truthful-in-expectation mechanism? Since exact optimization of a randomly perturbed objective does not generally yield a truthful mechanism, we require another idea.

7.4 Adjoint Perturbations

We now narrow the discussion to *linear* perturbations for non-negative linear maximization problems, where $\Psi(\Omega, v) = Pv$ for a (random) matrix P whose distribution is independent of v. We next develop a "duality" for such perturbations. We will need both the "primal" and "dual" viewpoints to prove the running time bound and the truthfulness guarantee, respectively, of our final mechanism.

Here is a trivial observation: for every fixed perturbation matrix P, objective function v, and feasible solution $x \in \Omega$, the value $(Pv)^T x$ of the solution x with respect to the perturbed objective Pv equals the value $v^T(P^T x)$ of the "perturbed solution" $P^T x$ with respect to the true objective v. We say that the perturbation P^T is *adjoint* to P. Taking this alternative adjoint viewpoint, solving a linearly perturbed instance (Ω, Pv) of a linear maximization problem is equivalent to solving

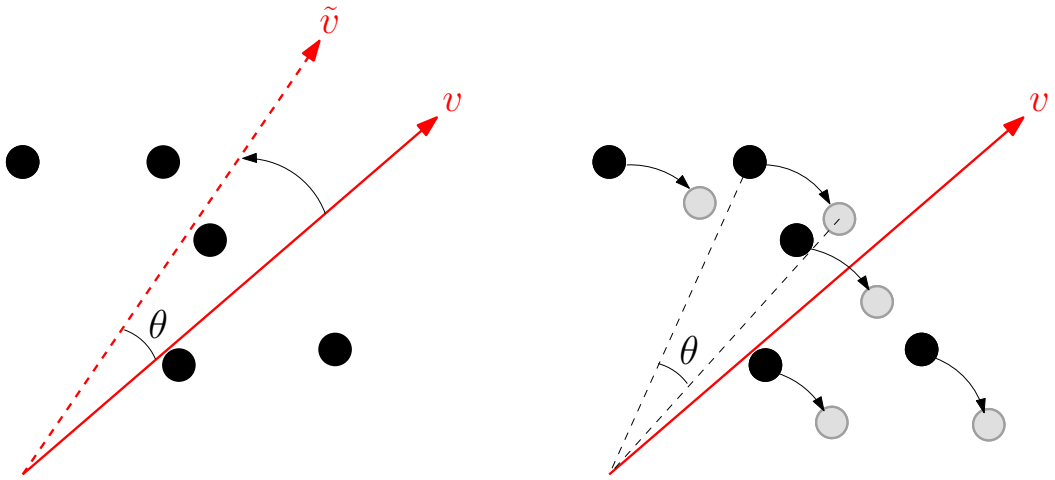
$$\begin{array}{ll} \text{maximize} & v^T \widetilde{x} \\ \text{subject to} & \widetilde{x} \in P^T\Omega, \end{array} \tag{7.1}$$

where $\widetilde{x} = P^T x$ and $P^T\Omega = \{\widetilde{x} : x \in \Omega\}$. See Figure 7.1 for an example of this relationship.

The adjoint problem (7.1) is meaningful when we can associate every $\widetilde{x} \in P^T\Omega$ with a probability distribution over the feasible solutions Ω that has expectation $\widetilde{x}$. This is possible if and only if $P^T\Omega \subseteq \text{convexhull}(\Omega)$. Assume that we have designed P to possess this property, and for every $x \in \Omega$ let D_x be an arbitrary distribution over Ω with expectation $\widetilde{x} = P^T x$. Let $\mathcal{R} = \{D_x\}_{x \in \Omega}$ denote the corresponding distributional range. By linearity, the adjoint problem (7.1) is then equivalent to the problem of maximizing the expected objective function value over $\mathcal{R}$:

$$\begin{array}{ll} \text{maximize} & \mathbf{E}_{y \sim D_x}[v^T y] \\ \text{subject to} & D_x \in \mathcal{R}. \end{array} \tag{7.2}$$

The key point is that *this is precisely the type of optimization problem that can be solved — truthfully — using an MIDR allocation rule and the corresponding payment rule* (recall Chapter 3).



(a) A primal perturbation rotating the objective by an angle of θ.

(b) The adjoint perturbation rotating feasible points by an angle of $-\theta$.

Figure 7.1: Duality of perturbations.

7.5 The Perturbation-Based Allocation Rule

The next theorem formalizes our progress so far: designing truthful-in-expectation mechanisms reduces to designing perturbations that meet a number of requirements.

For a linear perturbation Ψ for a non-negative linear maximization problem Π, we say that Ψ is *feasible* if, for every feasible set Ω of Π, the perturbation matrix $P \sim \Psi(\Omega)$ satisfies $P^T\Omega \subseteq \text{convexhull}(\Omega)$ surely. Such a perturbation is *tractable* if the perturbation matrix $P \sim \Psi$ can be sampled in time polynomial in the length of the description of Ω; and if for each $x \in \Omega$ a distribution D_x with support Ω and expectation $P^T x$ can be sampled in time polynomial in the length of the description of Ω. An *ϵ-FLAT* perturbation is one that is feasible, linear, ϵ-approximation preserving, and tractable. The outline of our black-box reduction is displayed below as Algorithm 7.1.

Theorem 7.5.1. *For every non-negative linear maximization problem Π and ϵ-FLAT perturbation Ψ for some $\epsilon \geq 0$, the corresponding perturbation-based (PB) allocation rule (Algorithm 7.1) satisfies the following properties:*

Parameter: Exact algorithm $\mathcal{A}$ for Π.
Parameter: ϵ-FLAT perturbation Ψ for Π.
Input: Instance (Ω, v).
Output: Solution $y \in \Omega$
1: Draw $P \sim \Psi(\Omega)$, and let $x = \mathcal{A}(\Omega, Pv)$.
2: Let D_x be a distribution over Ω with expectation $P^T x$.
3: Return a sample $y \sim D_x$.

Algorithm 7.1: The perturbation-based (PB) allocation rule.

(a) it is MIDR, and hence defines a truthful-in-expectation mechanism when combined with suitable payments;

(b) for every instance of Π*, it outputs a feasible solution with expected objective function value at least* $(1 - \epsilon)$ *times the maximum possible;*

(c) its worst-case expected running time is bounded by a polynomial plus that of the exact algorithm $\mathcal{A}$ *on a perturbed instance* (Ω, Pv)*.*

The key point of Theorem 7.5.1 is part (a), which guarantees truthfulness while permitting remarkable freedom in designing perturbations.

Proof of Theorem 7.5.1. First, we note that the choice of P in step 1 is independent of v by the definition of a linear perturbation and step 3 is well defined because Ψ is feasible. Part (c) follows immediately from the assumption that Ψ is tractable. Part (b) follows from the definition of an approximation-preserving perturbation, the fact that $\mathcal{A}$ is an exact algorithm, and the fact that the expected value of the solution y returned by the PB allocation rule equals $\mathbf{E}_{y \sim D_x}[v^T y] = v^T(P^T x) = (Pv)^T x$, which is the objective function value (with respect to the perturbed objective Pv) of the solution returned by $\mathcal{A}$.

To prove part (a), consider an instance (Ω, v). To begin, condition on the choice of P by $\Psi(\Omega)$ in step 1 of the PB allocation rule. Let D_x be the distribution over Ω with expectation $P^T x$ that the allocation rule chooses in step 3 in the event that $x = \mathcal{A}(\Omega, Pv)$, and set $\mathcal{R} = \{D_x \, : \, x \in \Omega\}$. By the definition of this step, the range $\mathcal{R}$

depends only on Ω and is independent of the valuations v. Since the allocation rule explicitly computes the solution x^* that maximizes $(Pv)^T x$ over $x \in \Omega$ and then samples an outcome from the corresponding distribution D_{x^*}, and this x^* is the same solution that maximizes $\mathbf{E}_{y \sim D_x}[v^T y]$ over $x \in \Omega$ (i.e., over D_x in $\mathcal{R}$), the output of the allocation rule is the same (for each v) as that of the MIDR allocation rule with distributional range $\mathcal{R}$.

We have established that for each fixed choice of P, the PB allocation rule is an MIDR rule. Since the random choice of P is independent of the valuations v, the PB allocation rule is a probability distribution over MIDR rules. By Lemma 3.4.2, it is an MIDR allocation rule. □

Chapter 8

A Black Box Result

8.1 Introduction

Using the linear perturbation framework of Chapter 7, this chapter gives the first black-box reduction from arbitrary approximation algorithms to incentive-compatible approximation mechanisms for a non-trivial class of multi-parameter mechanism design problems. Specifically, we prove that every welfare maximization problem that admits an FPTAS and can be encoded as a packing problem satisfying some mild conditions also admits a truthful-in-expectation randomized mechanism that is an FPTAS.

Our result is possible because work in smoothed analysis has already established that problems in our class can be solved in expected polynomial time after a perturbation satisfying certain conditions is applied to their objective function. However, application of the perturbation framework of Chapter 7 requires perturbations to be linear, and moreover to "dualize" to a perturbation mapping each allocation to a lottery over allocations. We show that all these conditions can be simultaneously satisfied for our class of packing problems.

8.1.1 Summary of Results and Techniques

The main question of algorithmic mechanism design, discussed in the introduction to this thesis, asks whether incentive-compatible efficient computation is fundamentally less powerful than "classical" efficient computation. This question remains poorly understood. A starry-eyed mechanism designer might hope for the best-possible answer: polynomial-time algorithms can be supplemented with incentive compatibility "without loss"; equivalently, if an optimization problem Π admits a polynomial-time α-approximation algorithm $\mathcal{A}$, then it admits a polynomial-time α-approximate incentive-compatible mechanism. Since such a result makes no hypotheses about the algorithm $\mathcal{A}$ beyond those on its running time and approximation factor, it would presumably be proved via a "black-box reduction" — a generic method that invokes $\mathcal{A}$ at most polynomially many times, and restores incentive compatibility without degrading $\mathcal{A}$'s approximation factor.

This chapter provides the first such black-box reduction for a non-trivial class of multi-parameter mechanism design problems. We consider a natural sub-class of the linear maximization problems considered in Chapter 7. When such a problem encodes welfare maximization, and moreover admits a (non-incentive-compatible) fully polynomial time approximation scheme (FPTAS)[1], we show that it also admits a truthful-in-expectation randomized mechanism that is an FPTAS.

We now elaborate on the approach for the main result of this chapter. Consider a linear maximization problem Π that admits an FPTAS, and has a polynomial number of decision variables. Applying a result of Röglin and Teng [79], Π admits an exact algorithm $\mathcal{A}$ with *polynomial smoothed complexity*. Informally this means that, given an instance (Ω, v) of Π and a "sufficiently random" noise vector δ, the expected runtime of $\mathcal{A}$ on a perturbed instance $(\Omega, v + \delta)$ is polynomial. (See Section 8.2.2 for precise definitions.) This is good news, as the linear perturbation machinery of Chapter 7 allows the application of perturbations in a way that preserves incentive

[1]Recall that a *fully polynomial time approximation scheme (FPTAS)* for a maximization problem takes as input an instance and an approximation parameter ϵ, and returns a feasible solution with objective function value at least $1-\epsilon$ times that of an optimal solution, in time polynomial in the size of the instance and in $1/\epsilon$. For randomized algorithms, the running time bound and approximation guarantee should hold, for every input, in expectation over the random coin flips of the algorithm.

compatibility. The bad news is that we must simultaneously satisfy the conditions for polynomial smoothed complexity, as well the conditions of Chapter 7 — feasibility, linearity, approximation-preservation, and tractability. This is not obviously possible a-priori, and appears impossible in general. Nevertheless, our main result shows that such perturbations exist for *binary packing problems*: those linear maximization problems where the feasible set Ω is encoded as a set of binary vectors, and moreover Ω is downward-closed (i.e., if $x \in \Omega$ and $y \leq x$ component-wise, then $y \in \Omega$). We note that since we use perturbations only as an algorithmic tool internal to our algorithm, we bear no burden of ensuring that the perturbations are "natural" in any sense (unlike in traditional smoothed analysis).

We also extend the main result of this chapter in various ways, including to binary covering problems in Section 8.6.1 and to non-packing binary maximization problems in Section 8.6.2.

8.1.2 Related Work

There are two known black-box reductions from truthful-in-expectation mechanism design to approximation algorithm design, and both are for *single-parameter* mechanism design problems (See Chapter 10). The space of truthful mechanisms for single-parameter problems is well understood and reasonably forgiving: an approximation algorithm can be used in a truthful mechanism if and only if it is *monotone* (Chapter 10). The first black-box reduction is due to Briest et al. [16], who proved that every single-parameter binary maximization problem that admits an FPTAS also admits a truthful mechanism that is an FPTAS. Their black-box reduction is also deterministic. Second, Babaioff et al. [7] exhibit a black-box reduction that converts an approximation algorithm for a single-parameter problem to a truthful mechanism. However, their reduction degrades the approximation ratio by a super-constant factor. Both of these black-box reductions rely heavily on the richness of the monotone algorithm design space, and do not admit obvious extensions to multi-parameter problems.[2]

[2]For example, the black-box reduction in [16] uses a simple truncation trick that preserves monotonicity but violates the weak monotonicity condition needed for truthfulness in multi-parameter problems.

For multi-parameter problems, the result of Lavi and Swamy [60] is in the spirit of black-box reductions. They show how to convert certain approximation algorithms to truthful-in-expectation mechanisms without degrading the approximation ratio. However, their result imposes non-trivial extra requirements on the approximation algorithm that is to be converted into a truthful approximation mechanism. For many problems, it is not clear if there are near-optimal approximation algorithms that meet these extra requirements.

On the negative side, here is no general and lossless black-box reduction from approximation algorithms to deterministic truthful approximation mechanisms for multi-parameter problems. This fact was first established by Lavi et al. [59], and Papadimitriou et al. [74] gave a quantitatively much stronger version of this impossibility result. Very recently, Dobzinski [34] proved a strong impossibility result for a variant of combinatorial public projects, showing in the process that there is no general and lossless black-box reduction from approximation algorithms to truthful-in-expectation approximation mechanisms.

There are more general black-box reductions in the more permissive Bayesian setting, where player valuations are drawn from a known prior. The first such result is due Hartline and Lucier [48], who show that for *every* single-parameter welfare maximization problem, every approximation algorithm can be made Bayesian incentive-compatible without degrading the expected approximation factor. Very recently, this result was extended to multi-parameter problems, under some conditions on the Bayesian prior, by Bei and Huang [8] and Hartline et al. [47].

Finally, we know of only one previous application of smoothed analysis techniques to the design of new algorithms: Kelner and Spielman [53] used an iterative perturbation approach to design a randomized simplex-type algorithm that has (weakly) polynomial expected running time.

8.2 Model and Preliminaries

8.2.1 Binary Packing Problems

A *binary maximization problem* is a non-negative linear maximization problem (Section 7.2) whose feasible set Ω consists of d-dimensional binary vectors. Formally, an instance of a binary maximization problem is given by feasible set $\Omega \subseteq \{0,1\}^d$ and non-negative vector $v \in \mathbb{R}^d_+$ of coefficients, and the goal is to compute a feasible solution $x \in \Omega$ that maximizes the linear objective $v^T x$. A *binary packing problem* is a binary maximization problem with additional structure in the feasible set Ω: if x belongs to Ω, and $y_i \leq x_i$ for all i, then $y \in \Omega$ as well. (Binary covering problems can be defined analogously; see Section 8.6.1.)

We are mainly interested in *welfare-maximization binary packing problems*, where the objective function $v^T x$ is the welfare of self-interested players with private valuation functions. Specifically, $v = \sum_i u_i$, where u_i is the valuation function of player i. The knapsack allocation problem (Example 7.2.1) is an example of such a problem. The knapsack allocation problem has *polynomial dimension*, meaning that the number d of decision variables is polynomial in the size of the description of the feasible set. The results of this chapter are for problems with polynomial dimension, but some of the techniques can be adapted to some interesting problems with exponential dimension – see Chapter 9.

8.2.2 Smoothed Complexity Basics

Smoothed complexity was defined by Spielman and Teng [83]; our formalism is similar to that in Beier and Vöcking [9] and Röglin and Teng [79]. A *perturbed instance* of a binary packing problem Π consists of a fixed feasible set $\Omega \subseteq \{0,1\}^d$ and d random variables $v_1, \ldots, v_d$, where each v_i is drawn independently from a distribution with support in $[0, v_{\max}]$ and a density function that is bounded above everywhere by $\phi/v_{\max}$. The parameter ϕ measures the maximum concentration of the distributions of the v_i's. We say that an algorithm $\mathcal{A}$ for a binary packing problem Π runs in

smoothed polynomial time if its expected running time is polynomial in the description length of Ω and ϕ for every perturbed instance.

Our work relies on the fact that every FPTAS for a binary maximization problem with polynomial dimension can be converted into an algorithm that runs in smoothed polynomial time. This is a special case of a result of Röglin and Teng [79], who strengthen a result of Beier and Vöcking [9].

Proposition 8.2.1 ([9, 79])**.** *For every FPTAS $\mathcal{F}$ for a binary maximization problem Π of polynomial dimension, there is an exact algorithm $\mathcal{A}^{\mathcal{F}}$ for Π that runs in smoothed polynomial time.*

Moreover, the quite natural algorithm $\mathcal{A}^{\mathcal{F}}$ in Proposition 8.2.1 treats $\mathcal{F}$ as an "oracle" or "black box", meaning that its behavior depends only on the outputs of $\mathcal{F}$ and not on the actual description of $\mathcal{F}$.[3]

8.2.3 Perturbation Schemes

A *perturbation scheme* Ψ for an optimization problem Π is a family of perturbations for Π (as defined in Section 7.2), parametrized by $\epsilon > 0$. We use Ψ_ϵ to denote the perturbation of scheme Ψ with parameter ϵ. A perturbation scheme Ψ is *approximation preserving* if Ψ_ϵ is an ϵ-approximation preserving perturbation (Section 7.2) for every parameter $\epsilon > 0$. Similarly, Ψ is feasible/linear/tractable if Ψ_ϵ is feasible/linear/tractable (Chapter 7) for each $\epsilon > 0$. Finally, perturbation scheme Ψ is *FLAT* if it is feasible, linear, approximation preserving, and tractable.

8.3 The Random Singleton Scheme

We now describe a FLAT perturbation scheme that leads to the main result of this chapter: every binary packing problem with polynomial dimension that admits an

[3]The results in [9, 79] are stated as conversions from randomized pseudopolynomial-time algorithms to smoothed polynomial-time algorithms. Proposition 8.2.1 follows since every FPTAS for a binary optimization problem of polynomial dimension can be converted easily to a pseudopolynomial-time exact algorithm in a black-box manner.

FPTAS also admits a truthful-in-expectation mechanism that is an FPTAS. We call our FLAT scheme the *random singleton (RS)* perturbation scheme, and describe it first via its adjoint (see Section 7.4) in Algorithm 8.1. The fact that each δ_i chosen in Step 1 is at most ϵ/d implies that the probabilities of Step 2 sum to 1. Also, because Ω is the feasible set of a binary packing problem, the all-zero vector lies in Ω, and we can assume without loss of generality that each basis vector $e_1, \ldots, e_d$ lies in Ω (if $e_i \notin \Omega$ then we can ignore coordinate i). Therefore, the solution y output by Algorithm 8.1 is always feasible.

Parameter: $\epsilon > 0$.
Input: Feasible set $\Omega \subseteq \{0,1\}^d$ of a binary packing problem.
Input: $x \in \Omega$
Output: $y \in \Omega$

1: For each $i = 1, 2, \ldots, d$, draw δ_i uniformly from the interval $[0, \epsilon/d]$;
2: Output a random solution $y \in \Omega$ according to the following distribution:

- x with probability $1 - \epsilon$;
- the "singleton" e_j with probability $(\sum_{i=1}^{d} \delta_i x_i)/d$ (for each $j = 1, \ldots, d$);
- the all-zero solution with the remaining probability.

Algorithm 8.1: Adjoint of the random singleton (RS) perturbation scheme.

The motivation of the random choices in Step 1 of Algorithm 8.1 is to ensure that the distribution defined by the perturbation is diffuse enough to permit algorithms with polynomial smoothed complexity (cf., the parameter ϕ in Section 8.2.2). The motivation of the random choices in Step 2 is to reward a solution $x \in \Omega$ with a "bonus" of a random singleton with probability δ_i for each coordinate i with $x_i = 1$. Since there exists a singleton e_j with value v_j that is at least a $1/d$ fraction of the optimal value $\max_{y \in \Omega} v^T y$, these bonuses effectively ensure that the perturbations occur at the correct "scale."

We now make this vague intuition precise. After conditioning on the random choices in Step 1 of Algorithm 8.1, the expectation $\widetilde{x}$ of the distribution D_x over

solutions Ω defined by Step 2 can be expressed via the adjoint perturbation P^T given by

$$\widetilde{x} = P^T x = (1-\epsilon)x + \left(\sum_{i=1}^{d} \delta_i x_i\right)\left(\sum_{j=1}^{d} \frac{e_j}{d}\right) \tag{8.1}$$

Let δ denote the d-vector of δ_i's. Since P^T can be written as $(1-\epsilon)I + \frac{1}{d}\vec{\mathbf{1}}\delta^T$, dualizing gives the following primal form of the RS perturbation:

$$P = (1-\epsilon)I + \frac{\delta \vec{\mathbf{1}}^T}{d}. \tag{8.2}$$

This corresponds to the linear perturbation given by the map

$$v_i \mapsto (1-\epsilon)v_i + \frac{\delta_i}{d}\sum_{j=1}^{d} v_j \tag{8.3}$$

for each coefficient i. We summarize the resulting (primal) form of the RS perturbation scheme in Algorithm 8.2. This perturbation scheme adds "bonuses" that depend on reported v_i's, and consequently might appear unsuitable for deployment in a truthful mechanism. But its use is justified by our development of adjoint perturbations.

Parameter: $\epsilon > 0$.
Input: Feasible set $\Omega \subseteq \{0,1\}^d$ of a binary packing problem.
Input: $v \in \mathbb{R}^d_+$
Output: $\widehat{v} \in \mathbb{R}^d_+$

1: For each $i = 1, 2, \ldots, d$, draw δ_i uniformly from the interval $[0, \epsilon/d]$;
2: Let

$$\widehat{v_i} = (1-\epsilon)v_i + \frac{\delta_i}{d}\sum_{j=1}^{d} v_j$$

for each $i = 1, \ldots, d$.

Algorithm 8.2: The random singleton (RS) perturbation scheme.

Lemma 8.3.1. *For every binary packing problem* Π *of polynomial dimension, the RS perturbation scheme (Algorithm 8.2) is FLAT.*

Proof. Since the choice of the perturbation matrix P (Equation (8.2)) depends only on the approximation parameter ϵ, the feasible set Ω, and the (valuation-independent) choices of the δ_i's, the RS scheme is linear. It is feasible because it is defined explicitly via its adjoint (Algorithm 8.1), which, for each $x \in \Omega$, outputs a distribution D_x with expectation equal to $P^T x$ (Equation (8.1)). It is clearly tractable. Finally, we observe from equation (8.3) that for every perturbation matrix P of the scheme and feasible solution $x \in \Omega$, $(1-\epsilon)v^T x \leq (Pv)^T x \leq v^T x + \epsilon \max_i v_i$. Since both $v^T x$ and $\max_i v_i$ are no greater than the value of the optimum solution, the RS scheme is approximation preserving. □

8.4 The Black Box Result

We are now prepared to prove the main result of this chapter.

Theorem 8.4.1. *Every welfare-maximization binary packing problem of polynomial dimension that admits an FPTAS also admits a truthful-in-expectation mechanism that is an FPTAS.*

Proof. Let Π be a binary packing problem of polynomial dimension and $\mathcal{F}$ and arbitrary FPTAS for it. By Proposition 8.2.1, there is an exact algorithm $\mathcal{A}^{\mathcal{F}}$ for Π that runs in smoothed polynomial time in the sense of Section 8.2.2. Let Ψ denote the RS perturbation scheme for Π, as given in Algorithm 8.2. For each approximation parameter ϵ, we instantiate the PB allocation rule (Algorithm 7.1) with the perturbation Ψ_ϵ and algorithm $\mathcal{A}^{\mathcal{F}}$. Since Ψ_ϵ is ϵ-FLAT (Lemma 8.3.1), Theorem 7.5.1 implies that this allocation rule is MIDR, has an approximation guarantee of $1-\epsilon$ in expectation, and has expected running time bounded by a polynomial plus that of $\mathcal{A}^{\mathcal{F}}$ on the perturbed instance (Ω, Pv).

To analyze the expected running time of $\mathcal{A}^{\mathcal{F}}$ on (Ω, Pv), recall the perturbation formula (8.3) of the RS scheme. Let $v_{\max}$ denote $\max_{i=1}^d v_i$. Every coordinate of Pv is bounded above by $v_{\max}$ with probability 1, and these coordinates are independent random variables (since the δ_i's are independent). Since $\sum_{j=1}^d v_j \geq v_{\max}$ and δ_i is drawn uniformly from $[0, \epsilon/d]$, the density of the random variable $(Pv)_i$ is bounded

above everywhere by $\frac{d^2}{\epsilon v_{\max}}$. Thus the concentration parameter ϕ from Section 8.2.2 is bounded by d^2/ϵ. Since Π has polynomial dimension and $\mathcal{A}^{\mathcal{F}}$ has polynomial smoothed complexity, the expected running time of $\mathcal{A}^{\mathcal{F}}$ on (Ω, Pv) is polynomial in the input size and $1/\epsilon$.

Finally, to complete the proof we note that Proposition 3.4.1 implies that truth-telling payments for this allocation rule can be computed with poly(n) overhead in runtime. □

8.5 Example Problems

We feel that the primary point of Theorem 8.4.1 is conceptual: it shows that requiring (randomized) incentive compatibility requires no sacrifice in performance for a non-trivial class of multi-parameter problems, and suggests that even more general "black-box randomized reductions" might be possible. Of course, a general result like Theorem 8.4.1 can be instantiated for various concrete problems, we list a few examples. Numerous single-parameter examples are given in Briest et al. [16]. Below we present some multi-parameter examples, which are beyond the reach of the results in [16].

Knapsack Allocation: From a purely algorithmic perspective, the problem in Example 7.2.1 is equivalent to the knapsack problem and hence admits a (non-truthful) FPTAS.

Arborescent Knapsack Allocation: This is a generalization of the knapsack allocation problem, where additional constraints are placed on the feasible solutions $\Omega \subseteq \{0,1\}^m$. Namely, the projects $[m]$ are the ground set of a laminar[4] set system $\mathcal{L} \subseteq 2^{[m]}$, and there is a budget C_T for each $T \in \mathcal{L}$. The feasible set Ω is constrained so that $\sum_{j \in T} s_j \leq C_T$ for each $T \in \mathcal{L}$. A (non-truthful) FPTAS for this problem was given in [45].

Tree-Ordered Knapsack Allocation: This is another generalization of the knapsack allocation problem, where precedence constraints are placed on the projects

[4]A set system $\mathcal{L} \subseteq 2^{[m]}$ is *laminar* if for each $T, T' \in \mathcal{L}$ either $T \cap T' = \emptyset$, or $T \subseteq T'$, or $T' \subseteq T$.

$[m]$. Namely, a directed acyclic graph G with vertices $[m]$ encodes precedence constraints, and the feasible set $\Omega \subseteq \{0,1\}^m$ is constrained so that $x_j \geq x_k$ whenever $(j,k) \in E(G)$ for every $x \in \Omega$. When G is a directed-out tree, a (non-truthful) FPTAS for this problem was given in [52]. Observe, however, that this is no longer a binary packing problem. Fortunately, our proof of Theorem 8.4.1 relied very little on the packing assumption: we argue in Section 8.6.2 that we only require $\vec{\mathbf{0}} \in \Omega$, which is certainly the case here.

Maximum Job Sequencing with Deadlines: In this problem, m jobs are to be scheduled on a single machine. Job $j \in [m]$ has processing time p_j and deadline d_j. There are n players, and player i has private value u_{ij} for each job j that completes before its deadline d_j. The goal is to find the welfare-maximizing subset of the jobs that can be scheduled so that each finishes before its deadline. Converting such a set of jobs to a schedule can be done via the obvious greedy algorithm. This yields a binary packing problem with a welfare objective. A (non-truthful) FPTAS for this problem was given in [80].

8.6 Extensions

8.6.1 Binary Covering Problems

In this section, we consider *binary covering problems* of polynomial dimension. Such problems are defined analogously to binary packing problems, except that the feasible Ω is upward closed and the goal is to minimize $v^T x$ over $x \in \Omega$. We consider social-cost-minimization binary covering problems, where $v^T x$ is the social cost of outcome x. In these problems, $v_j = \sum_{i=1}^{n} c_{ij}$ for each $j = 1, 2, \ldots, d$, and c_i denotes the private cost function of player i.

By reduction to Theorem 8.4.1, we show that binary covering problems admit a truthful-in-expectation "additive FPTAS". We then show that this is the best we can hope for by an MIDR mechanism, as no polynomial-time MIDR mechanism obtains a finite approximation for an NP-hard binary covering problem (assuming $P \neq NP$).

Theorem 8.6.1. *Every social-cost-minimization binary covering problem of polynomial dimension that admits an FPTAS also admits a truthful-in-expectation mechanism, parametrized by ϵ, that runs in time polynomial in the description of the instance and $\frac{1}{\epsilon}$, and outputs a solution with expected social cost at most an additive $\epsilon c_{\max}$ more than the optimum value, where $c_{\max}$ denotes the maximum cost of a feasible solution.*

Proof. Let Π be a binary covering problem of polynomial dimension d that admits an FPTAS. We now define the *complementary problem* $\overline{\Pi}$ as follows. For $x \in \{0,1\}^d$, we define its complement $\overline{x} = \vec{\mathbf{1}} - x$. For $\Omega \subseteq \{0,1\}^d$ let $\overline{\Omega} = \{\overline{x} : x \in \Omega\}$. Now let $\overline{\Pi} = \left\{(\overline{\Omega}, v) : (\Omega, v) \in \Pi\right\}$ be the problem of maximizing $v^T\overline{x}$ for $\overline{x} \in \overline{\Omega}$. It is clear that $\overline{\Pi}$ is a binary packing problem of polynomial dimension. It is easy to see that x is an optimal solution for Π if and only if $\overline{x}$ is an optimal solution to $\overline{\Pi}$. We utilize this complementarity between covering and packing problems twice — once in each direction — in the following proof: first we argue that an FPTAS for Π yields an FPTAS for $\overline{\Pi}$, then we invoke the result in Theorem 8.4.1 to obtain an MIDR FPTAS for $\overline{\Pi}$, which we then show can be converted to an MIDR "additive FPTAS" for Π. Invoking Proposition 3.4.1 then completes the proof.

Now we argue that an FPTAS for Π can be converted to an FPTAS for $\overline{\Pi}$ as follows: for an instance $(\overline{\Omega}, v)$ of $\overline{\Pi}$ and approximation parameter ϵ, we simply invoke the FPTAS for Π on (Ω, v) with approximation parameter ϵ/d, and output the complement of the returned solution. Let OPT denote the optimal objective function value of the covering problem Π on instance (Ω, v). The cost of the solution x returned by the FPTAS for Π is within an additive error of at most $(\epsilon/d)OPT \leq (\epsilon/d) \cdot dv_{\max} = \epsilon v_{\max}$ from optimal, where $v_{\max} = \max_{i=1}^{n} v_i$. Since the optimal value of the complementary packing problem $\overline{\Pi}$ is — without loss of generality — at least $v_{\max}$, the value of $\overline{x}$ is is at least a $(1-\epsilon)$ factor of the optimum value for instance $(\overline{\Omega}, v)$ of the packing problem $\overline{\Pi}$.

By the result in Theorem 8.4.1, the packing problem $\overline{\Pi}$ admits an MIDR FPTAS $\overline{\mathcal{A}}$. We will now convert $\overline{\mathcal{A}}$ to an MIDR "additive FPTAS" $\mathcal{A}$ for the covering problem Π. We fix the approximation parameter ϵ, and define $\mathcal{A}$ as follows: On input (Ω, v), let $\overline{x}$ be the output of $\overline{\mathcal{A}}$ with approximation parameter ϵ/d and input $(\overline{\Omega}, v)$, and output $x = 1 - \overline{x}$.

To show that $\mathcal{A}$ is MIDR, let $\overline{R}$ be the range of $\overline{\mathcal{A}}$ when the approximation parameter is ϵ/d. Define the complementary range $\mathcal{R}$ in the obvious way: for every $\overline{D} \in \overline{\mathcal{R}}$, we let D be the distribution that simply draws $\overline{y} \sim \overline{D}$ and outputs $y = 1 - \overline{y}$. Then, we let $\mathcal{R} = \left\{D : \overline{D} \in \overline{\mathcal{R}}\right\}$. First, it is easy to see, by construction, that the range of $\mathcal{A}$ is a subset of $\mathcal{R}$. Now, fix an instance $(\Omega, v) \in \Pi$, and let $\overline{D}$ be the distribution of $\overline{\mathcal{A}}(\overline{\Omega}, v)$. By definition, $\mathcal{A}$ outputs the complementary distribution D for the same input. Since $\overline{D}$ maximizes $\mathbf{E}_{\overline{y}\sim\overline{D}}[v^T\overline{y}]$ over $\overline{\mathcal{R}}$, it must minimize $\mathbf{E}_{\overline{y}\sim\overline{D}}[v^T(\vec{\mathbf{1}} - \overline{y})]$ over $\overline{\mathcal{R}}$. This, by definition, implies that D minimizes $\mathbf{E}_{y\sim D}[v^T y]$ over $\mathcal{R}$, and $\mathcal{A}$ is MIDR with range $\mathcal{R}$.

It remains to prove the approximation guarantee of $\mathcal{A}$. Let $x \sim D$ and $\overline{x} \sim \overline{D}$ be the (random) outputs of $\mathcal{A}$ and $\overline{\mathcal{A}}$ on inputs (Ω, v) and $(\overline{\Omega}, v)$, respectively. We can bound the expected social cost as follows.

$$\begin{aligned}
\mathbf{E}[v^T x] &= \mathbf{E}[v^T(\vec{\mathbf{1}} - \overline{x})] \\
&= ||v||_1 - \mathbf{E}[v^T\overline{x}] \\
&\leq ||v||_1 - (1 - \epsilon/d) \max_{\overline{y}\in\overline{\Omega}} v^T\overline{y} \\
&= ||v||_1 - (1 - \epsilon/d)(||v||_1 - \min_{y\in\Omega} v^T y) \\
&\leq \min_{y\in\Omega} v^T y + \frac{\epsilon}{d}||v||_1 \\
&\leq \min_{y\in\Omega} v^T y + \epsilon \max_i v_i
\end{aligned}$$

Since $\max_i v_i$ is a lower bound on the maximum cost of solutions in Ω, this completes the proof. □

We note that the mechanism of Theorem 8.6.1, as stated in the proof, is not individually rational. This is because the Clarke pivot rule goes "in the wrong direction" for cost minimization problems, and consequently the payments of Proposition 3.4.1 violate individual rationality. It is natural to prefer a pivot rule that results in payments *to* the players, and moreover guarantees individual rationality. This is impossible for these cost minimization problems in general, but is possible for many interesting special cases. Notably, the class of binary covering problems where each

binary variable is "owned" by a single player, and only the variable's owner incurs a cost when that variable is set to 1, admits such payments when no player is a "monopoly": i.e. for each player i, there is a feasible solution that sets all player i's variables to 0. In a typical example, the set of solutions is the set of $s-t$ paths in a graph, the edges of the graph are partitioned between the players, and no player owns all edges of an s-t cut of the graph. Given an MIDR allocation rule $\mathcal{A}$ and reported costs c for such a problem, and letting $c'(i)$ denote the modification of c that sets player i's costs for his variables to ∞, the mechanism that pays each player i in the amount of $\mathbf{E}[\sum_{i' \neq i} c_{i'}(\mathcal{A}(c'(i))) - \sum_{i' \neq i} c_{i'}(\mathcal{A}(c))]$ is truthful in expectation and individually rational in expectation. An analogue of Proposition 3.4.1 easily follows, and we omit the details.

The bound in Theorem 8.6.1 becomes an FPTAS in the usual multiplicative sense when we restrict our attention to instances of the problem in which the value of the optimal solution can be bounded below by an inverse polynomial fraction of $c_{\max}$. In general, however, additive loss is inevitable if we restrict ourselves to MIDR algorithms.

Lemma 8.6.2. *Let Π be a binary minimization problem. If an MIDR algorithm $\mathcal{A}$ provides a finite approximation ratio for Π, then $\mathcal{A}$ is optimal.*

Proof. Assume $\mathcal{A}$ is MIDR, and provides a finite approximation ratio for Π. Fix a feasible set Ω of Π, and let $\mathcal{R}$ be the corresponding distributional range of $\mathcal{A}$. We say a feasible solution $x \in \Omega$ is *minimal* if there does not exist $y \neq x$ in Ω with $y_i \leq x_i$ for all i. It is clear that for every objective $v \in \mathbb{R}^d_+$, there exists an optimal solution that is minimal. Since $\mathcal{A}$ is MIDR, it then suffices to show that $\mathcal{R}$ contains all point distributions corresponding to minimal feasible solutions.

Consider a minimal $x \in \Omega$, and let the objective function v be such that $v_i = 0$ when $x_i = 1$, and $v_i = 1$ when $x_i = 0$. By definition we have $v^T x = 0$. Moreover, since x is minimal, $v^T y > 0$ for every $y \in \Omega$ with $y \neq x$. Therefore, the only distribution over Ω providing a finite approximation ratio for v is the point distribution corresponding to x. Thus, $\mathcal{R}$ contains all point distributions of minimal feasible solutions, as needed. □

The following negative result for binary covering problems follows immediately from Lemma 8.6.2.

Theorem 8.6.3. *Let* Π *be an NP-hard binary minimization problem. No polynomial-time MIDR allocation rule provides a finite approximation ratio for* Π *unless* $P = NP$.

We note that Theorem 8.6.3 and its proof easily extend to the slightly more general class of *distributional affine maximizers* (see Section 3.4), and hence to all known types of VCG-based mechanisms.

Examples We conclude the section with a few multi-parameter problems to which Theorem 8.6.1, and the complementary negative result in Theorem 8.6.3, apply. Again, for numerous single-parameter examples see Briest et al. [16].

Minimum Job Sequencing with Deadlines: This is the minimization variant of Maximum Job Sequencing with Deadlines. Here, player i incurs a cost c_{ij} for every job j that completes past its deadline d_j. The goal is to minimize social cost. This is a binary covering problem. A (non-truthful) FPTAS for this problem was given in [44].

Constrained Shortest Path: We are given a graph $G = (V, E)$, and two terminals $s, t \in V$. Additionally, there is latency l_j for each $j \in E$. The mechanism is interested in selecting a path from s to t of total latency at most L. There are n players, and player i incurs private cost c_{ij} if $j \in E$ is selected. We consider a covering variant of this problem, where the mechanism may select any subgraph of G connecting s to t via a path of latency at most L, and the goal is to minimize social cost. A (non-truthful) FPTAS for this problem was given in [49].

Constrained Minimum Spanning Tree on Treewidth Bounded Graphs: We are given a graph $G = (V, E)$ with bounded treewidth. Additionally, there is a weight w_j for each $j \in E$. The mechanism is interested in selecting a spanning tree of G with total weight at most W. There are n players, and player i incurs private cost c_{ij} if $j \in E$ is selected. We consider the covering variant of this problem, where the mechanism may select any spanning subgraph of G containing a spanning tree of total weight at most W, and the goal is to minimize social cost. A (non-truthful) FPTAS for this problem was given in [62].

8.6.2 Non-Packing Binary Maximization Problems

We observe that the packing assumption of Theorem 8.4.1 can be relaxed. In particular, if Π is a binary maximization problem, it suffices that $\vec{\mathbf{0}} \in \Omega$ for every feasible set Ω of Π. To see this, observe that the only other property of packing problems that was used in the proof of Theorem 8.4.1 was that $e_j \in \Omega$ for each $j = 1, \ldots, d$. It is straightforward to modify the proof to use the following, relaxed, assumption: For each $j = 1, \ldots, d$, there exists $y^j \in \Omega$ such that $y^j_j = 1$ (and y^j can be identified in polynomial time). Letting $y = \sum_j y^j$, we then modify Ψ as follows: δ is drawn as before, and we let $P = (1 - \epsilon)I + \frac{\delta y^T}{d}$. The proof proceeds in a similar fashion. Similarly, Theorem 8.6.1 extends to binary minimization problems where $\vec{\mathbf{1}} \in \Omega$ for every feasible set Ω.

8.6.3 Stronger Guarantees on Payments

We note that the payments of Proposition 3.4.1, used for the result of Theorem 8.4.1, guarantee non-negative transfers and individual rationality only *in expectation*. It is desirable to strengthen this to a *universal*, or *ex-post* guarantee; i.e., to guarantee that player i's payment and utility are non-negative for each flip of the mechanism's coins. While this does not appear to be possible in polynomial time for an arbitrary MIDR allocation rule, we show in this section that it is possible for the PB allocation rule, due to its unique structure, when applied to a problem with polynomial dimension. Therefore, the result of Theorem 8.4.1 can be made to satisfy individual rationality and non-negative transfers universally.

We now how show how to compute the desired payments efficiently. As a first step, we invoke an observation from [60]: a payment rule guaranteeing ex-post individual rationality and non-negativity always *exists* for any MIDR mechanism for a problem with non-negative valuations. We summarize this observation for welfare maximization binary packing problems in Proposition 8.6.4, and include a proof for completeness. This payment rule carefully couples its random choices with the random choice of the accompanying allocation rule.

Proposition 8.6.4. *Fix a welfare-maximization binary packing problem, and let $\mathcal{A}$ be an MIDR allocation rule for this problem with distributional range $\mathcal{R}$. Let y be the random output of $\mathcal{A}$ on an arbitrary input u. The following payments — which depend on the outcome $y \sim \mathcal{A}(u)$ of the allocation rule — are non-negative, and result in a truthful-in-expectation and individually-rational mechanism when coupled with allocation rule $\mathcal{A}$.*

$$\overline{p}_i(u, y) = \frac{p_i^{vcg}(u)}{\mathbf{E}\left[u_i \cdot \mathcal{A}(u)\right]} \; u_i \cdot y, \tag{8.4}$$

where p_i^{vcg} is as defined in Equation (3.1)*.*

Proof. That $\overline{p}_i(u, y)$ is non-negative follows from the fact that the VCG payment $p_i^{vcg}(u)$ of player i is non-negative, as is his expected value $\mathbf{E}\left[u_i \cdot \mathcal{A}(u)\right]$ and his realized value $u_i \cdot y$. For truthfulness in expectation, we note that taking expectation over choices of y in Equation (8.4) recovers the VCG payment $p_i^{vcg}(u)$. Since using VCG payments yields a truthful-in-expectation mechanism, the same is true for the payments in Equation (8.4).

It remains to show individual rationality. Recall that the VCG payments in Equation (3.1) are individually rational in expectation — specifically $p_i^{vcg}(u) \leq \mathbf{E}[u_i \cdot \mathcal{A}(u)]$. This implies that the payment $\overline{p}_i(u, y)$ in Equation (8.4) is no greater than the realized value $u_i \cdot y$, as needed for individual rationality. □

Efficient computation of the payments of Proposition 8.6.4 is not possible in general. Fortunately, the PB allocation rule has additional structure that we can exploit to efficiently compute ex-post individually-rational and non-negative payments. The payments will be based on those of Proposition 8.6.4, and we describe them next.

We recall that the PB allocation rule (Algorithm 7.1) is MIDR for every choice of perturbation matrix P drawn from Ψ. Therefore, it suffices to compute the payments of Equation (8.4) for a fixed choice of P; this would yield a randomization over truthful-in-expectation mechanisms, one for each choice of P, each of which is ex-post individually rational and charges non-negative payments.

Fix ϵ, and let $\mathcal{B}$ denote an instantiation of the PB allocation rule with a smoothed polynomial-time algorithm $\mathcal{A}^{\mathcal{F}}$ and a FLAT perturbation scheme Ψ. Let $\mathcal{B}^P$ denote $\mathcal{B}$ when the perturbation matrix in Step 1 of Algorithm 7.1 is fixed to P. Let u_i be

the valuation vector of player i, so that $v = \sum_i u_i$. By examining equations (8.4) and (3.1) and invoking the polynomial dimensionality of our problem, we conclude that computing the payments of Equation (8.4) reduces in polynomial time to computing $\mathbf{E}[\mathcal{B}^P(v)]$ and $\mathbf{E}[\mathcal{B}^P(v-u_i)]$ for each player i. The expected outcome $\mathbf{E}[\mathcal{B}^P(v)]$ is $P^T x$, where $x = \mathcal{A}^{\mathcal{F}}(\Omega, Pv)$ is as computed in Step 1. Similarly, the expected outcome $\mathbf{E}[\mathcal{B}^P(v-u_i)]$ is $P^T\mathcal{A}^{\mathcal{F}}(\Omega, P(v-u_i))$. Since P is drawn from tractable perturbation scheme Ψ and $\mathcal{A}^{\mathcal{F}}$ runs in smoothed polynomial time, computing $\mathcal{A}^{\mathcal{F}}(\Omega, Pv)$ and $\mathcal{A}^{\mathcal{F}}(\Omega, P(v-u_i))$ for all players i takes expected time polynomial in the description of Ω, $\frac{1}{\epsilon}$ and the number of players.

Chapter 9

Multi-Unit Auctions

9.1 Introduction

In this chapter we consider welfare maximization in *multi-unit auctions*, where a large (exponential) number of identical items must be allocated among a set of competing players. This problem captures many scenarios where a homogeneous resource must be divided among multiple players, such as bandwidth, machine processing time, or electrical power. Multi-unit auctions admit a non-truthful FPTAS; however, despite intense study, no truthful mechanism with an approximation ratio better than 50% was known before our work.

This problem escapes the black-box result of Chapter 8 because its formulation as a linear optimization problem has an exponential number of variables. The results of smoothed analysis (Section 8.2.2) exploited in Chapter 8 are no longer applicable, and efficient application of linear perturbations (Chapter 7) is not in general possible for problems with exponential dimension. We overcome these difficulties to design a truthful-in-expectation FPTAS for multi-unit auctions: we design linear perturbations that exploit the structure of multi-unit auctions to recover polynomial smoothed complexity, and at the same time can be applied efficiently to solutions of exponential dimension using the principle of deferred decisions.

9.1.1 Summary of Results and Techniques

In multi-unit auctions, there are m identical items to be divided among n players. Each player's valuation is described by a function that maps a number between 0 and m to the player's value for receiving that number of items. We assume that each valuation is presented via a *value oracle*, meaning that the valuation may be queried efficiently at any number between 0 and m. We require no structure on these valuations besides them being non-decreasing (also known as free disposal) and normalized (the value of a player for no items is zero).

Welfare maximization in multi-unit auctions is possible in polynomial time, using dynamic programming, if runtime is allowed to depend polynomially on m. We instead consider the case where m is a large number of items, written in binary, and therefore our runtime must be polynomial in n and $\log m$. Under these restrictions, the problem generalizes the NP-hard knapsack problem.[1] Moreover, the FPTAS for the knapsack problem (see e.g. [84]) can be extended to multi-unit auctions. Given an approximation parameter ϵ, the FPTAS runs in time polynomial in n, $\log m$ and $\frac{1}{\epsilon}$, and returns a $(1-\epsilon)$-approximate solution to the welfare maximization problem in multi-unit auctions.

Application of the techniques of Chapters 7 and 8 to multi-unit auctions faces two main difficulties, both due to the exponential dimensionality of the problem. The first relates to our use of smoothed complexity; an FPTAS for a problem with exponential dimension does not appear to, in general, imply polynomial smoothed complexity of the problem in the same general sense defined in Section 8.2.2. We overcome this by designing a specific perturbation scheme that exploits the structure of multi-unit auctions, and an accompanying exact algorithm that solves instances perturbed by this scheme in expected polynomial time. Second, even if instances perturbed by a specific perturbation can be solved in expected polynomial time, this is only useful if said perturbation can be applied to an instance efficiently. This was called *tractability* in Chapter 7, and required efficient sampling of a perturbation matrix, as well as

[1] The knapsack problem is equivalent to the special case of multi-unit auctions where each player's valuation function is *single minded*, meaning that his valuation as a function of the number of items increases at most once.

efficient decomposition of a solution perturbed by the adjoint perturbation into a convex combination of other solutions. Both requirements are evidently impossible for problems of exponential dimension, as the perturbation matrix has an exponential number of entries. We relax tractability using the principle of deferred decisions, and show that the relaxed definition suffices for multi-unit auctions.

9.1.2 Related Work

Multi-unit auctions are as old as mechanism design. The problem was first considered by Vickrey [85], who studied the special case where players have downwards sloping valuations. The welfare-maximizing allocation can be computed efficiently in this case, and the VCG mechanism can be employed. The special case where players are *single-minded*, meaning that each bidder's valuation as a function of the number of items increases at most once, is NP-hard. Mu'alem and Nisan [67] exhibited a truthful polynomial-time 2-approximation mechanism for multi-unit auctions with single-minded bidders, and this was followed by a truthful FPTAS by Briest et al. [16]. The case where a player's valuation function increases at most k times, where k is polynomial in n and $\log m$, was considered by Lavi and Swamy [60], who exhibit a truthful-in-expectation 2-approximation mechanism, and this was strengthened to a PTAS by Dobzinski and Nisan [30]. For the general multi-unit auctions problem, a truthful-in-expectation 2-approximation mechanism was presented by Lavi and Swamy [60] in the demand oracle model,[2] followed by a deterministic 2-approximation mechanism in the value oracle model by Dobzinski and Nisan [30].

Negative results for polynomial-time mechanisms for multi-unit auctions began with the work of Lavi et al. [59], who showed that no deterministic truthful mechanism that always allocates all items can guarantee a 50% approximation ratio in polynomial time. Dobzinski and Nisan [30] proved the same 50% lower bound for deterministic VCG-based mechanisms, and in followup work [31] generalized their impossibility result to deterministic truthful mechanisms that are "scalable", in that they are independent of the units used to measure players' valuation functions.

[2] A demand oracle for a valuation v takes as input a price p, and returns the number of items j maximizing the player's utility $v(j) - pj$.

We note that the conference publication of this result ([27]) preceded the linear perturbation framework, and employed a different proof. The proof presented in this chapter uses the linear perturbation framework of Chapter 7, and we feel it is simpler and more instructive. Nevertheless, the mechanism in the conference publication is stronger in one sense: it runs in polynomial time surely, whereas the mechanism presented in this chapter runs in expected polynomial time.

After the conference publication of this result ([27]) and before publication of this thesis, Vöcking [86] showed that a perturbation approach, implemented differently, yields a universally-truthful FPTAS for multi-unit auctions. The approach of [86] bears some similarity to the approach presented in this chapter, and any common ideas were independently discovered.

9.2 Model and Preliminaries

9.2.1 Multi-Unit Auctions

In a multi-unit auction, a set of m identical items must be allocated among n bidders. Each bidder i is equipped with a valuation function $v_i : \{0, \ldots, m\} \to \mathbb{R}_+$, where v_i is non-decreasing, and normalized: $v_i(0) = 0$. The goal is to find an allocation $(s_1, \ldots, s_n)$ of the items, where each s_i is a non-negative integer and $\sum_{i=1}^{n} s_i \leq m$, that maximizes the social welfare: $\sum_i v_i(s_i)$. Multi-unit auctions can be written as a binary packing problem (Chapter 8) as follows, where decision variable x_{ij} indicates whether player i is assigned j items.

$$\begin{array}{lll} \text{maximize} & \sum_{ij} v_i(j) \cdot x_{ij} & \\ \text{subject to} & \sum_{ij} j \cdot x_{ij} \leq m & \\ & \sum_j x_{ij} \leq 1, & \text{for } i \in [n]. \\ & x_{ij} \in \{0, 1\}, & \text{for } i \in [n], j \in [m]. \end{array} \tag{9.1}$$

We assume that an instance of multi-unit auctions is given by the number of items m, and n valuation functions each presented via a *value oracle*. Specifically, for each player i and number of items j, our algorithms may query $v_i(j)$ in constant time.

Since all items are identical, the natural "length" of the description of the items is $\log m$, rather than m. Therefore, we require a computationally-efficient algorithm for multi-unit auctions to run in time polynomial in n and $\log m$.

Using n and $\log m$ as the natural parameters, we observe that the binary packing problem (9.1) is of exponential dimension — specifically, it has mn variables. Therefore, the main result of Chapter 8 does not directly apply to multi-unit auctions.

9.3 Outline of our Approach

As mentioned in the introduction, application of the techniques of Chapters 7 and 8 to multi-unit auctions faces two main difficulties, both due to the exponential dimensionality of the problem. We now outline these challenges, and our approach to overcoming them, in more detail.

Smoothed Polynomial Runtime with Exponential Dimension The PB allocation rule (Algorithm 7.1) requires the existence of an exact algorithm for multi-unit auctions that runs in expected polynomial time over perturbed instances. Unfortunately, the results of smoothed analysis (Section 8.2.2) do not hold in general for problems with exponential dimension. Nevertheless, we will show that there is a specific perturbation scheme and accompanying exact algorithm, both designed carefully to exploit the particular structure of multi-unit auctions, such that the expected runtime of the algorithm over perturbed instances is polynomial in n and $\log m$. We now describe the intuition behind our perturbation scheme.

We start with the following observation. In searching for an optimal solution to an instance of multi-unit auctions, it suffices to evaluate a valuation function v of a player only at *dominant points*, defined as follows: A point $k \in [m]$ is *dominant* in valuation function $v : [m] \to \mathbb{R}_+$ if $v(k) > v(j)$ for all $j < k$. If we restrict our attention to instances of multi-unit auctions with a polynomial number of dominant points in each player's valuation function, and moreover assume that a list of these points is provided as part of the input to the problem, we could re-write an instance

as a binary packing problem with *polynomial dimension*, and apply the result of Chapter 8 to obtain a truthful-in-expectation FPTAS for these instances.

No such guarantee on the number of dominant points is possible in general. Fortunately, however, we will show that there is a specific linear perturbation of valuation functions, which we refer to as the *2-adic* perturbation, such that the perturbed function is guaranteed to have only a polynomial number of dominant points, and moreover a list of these points can be computed in polynomial time. Composing the 2-adic perturbation with a variant of the RS perturbation scheme (Section 8.3) will then yield a truthful-in-expectation FPTAS as in Chapter 8.

Tractable Perturbation with Exponential Dimension Polynomial-time solvability of the perturbed optimization problem does not by itself suffice. We must additionally apply our perturbation scheme efficiently. In Chapter 7, this requirement was called *tractability*, and required efficient sampling of a perturbation matrix P, as well the construction of a perturbed solution with expectation $P^T x$ given a feasible solution x. Unfortunately, this requirement is a non-starter for problems of exponential dimension; this is because no scheme can produce in polynomial time an explicit description of a perturbation matrix with an exponential number of entries. Nevertheless, we observe that a relaxed definition of tractability suffices. We say a perturbation scheme Ψ for multi-unit auctions is *tractable with deferred decisions* if it satisfies the following three properties:

(I) A matrix $P \in \mathbb{R}^{n \times m}$ drawn from Ψ is uniquely determined by a vector δ of independent random variables. We require that the length of δ is at most exponential in n and $\log m$ so that individual entries of δ may be indexed using polynomial space. Moreover, we requires that each entry of δ can be sampled efficiently.

(II) For each objective vector $v \in \mathbb{R}^{n \times m}$ for multi-unit auctions and each $i \in [n]$ and $j \in [m]$, an entry $(Pv)_{ij}$ of the perturbed valuation vector can be evaluated in polynomial time by reading only a polynomial (in n and $\log m$) number of entries of δ.

(III) For each binary vector x in the feasible set Ω for multi-unit auctions, represented succinctly as a mapping from player to number of items, a distribution D_x with support Ω and expectation $P^T x$ can be sampled in time polynomial in n and $\log m$ by reading only a polynomial number of entries of δ.

We will design our perturbation scheme to satisfy tractability with deferred decisions. To account for this modified notion of tractability, we won't require explicit computation of P in Step 1 of the Perturbation Based (PB) allocation rule (Algorithm 7.1). In the *modified PB allocation rule*, we modify Step 1 of Algorithm 7.1 as follows: we give algorithm $\mathcal{A}$ oracle access to both v and the vector δ of random variables describing the perturbation matrix P, and require that $\mathcal{A}$ output an optimal solution to the perturbed instance (Ω, Pv). The following modification of Theorem 8.4.1 now sets the stage.

Theorem 9.3.1. *If Ψ is a feasible, linear, approximation-preserving perturbation scheme for multi-unit auctions, and moreover Ψ is tractable with deferred decisions, then the modified perturbation-based (PB) allocation rule for multi-unit auctions satisfies the following properties:*

(a) it is MIDR, and hence defines a truthful-in-expectation mechanism when combined with suitable payments;

(b) for every approximation parameter $\epsilon > 0$ and instance of Π, it outputs a feasible solution with expected objective function value at least $(1-\epsilon)$ times the maximum possible;

(c) its worst-case expected running time is bounded by a polynomial in n and $\log m$ plus the runtime of the exact algorithm $\mathcal{A}$ for applying the perturbation and producing the exact solution to the perturbed instance (Ω, Pv).

Proof. The proof of (a) and (b) is identical to that in Theorem 8.4.1. For (c), note that Properties (I) and (III) of schemes that are tractable with deferred decisions imply that Steps 2 and 3 of the PB allocation rule (Algorithm 7.1) can be implemented in polynomial time. □

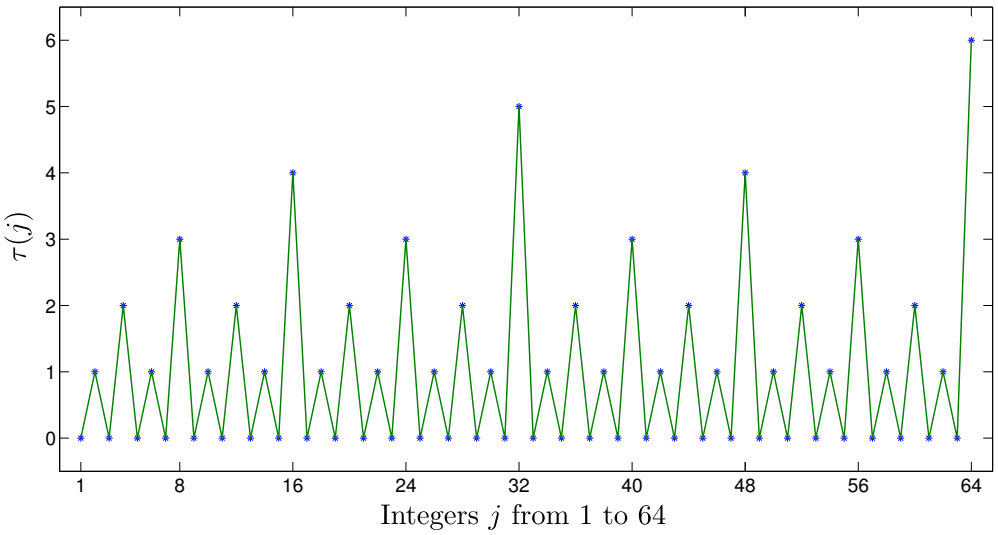


Figure 9.1: The 2-adic valuation.

9.4 The 2-adic Perturbation

In this section, we define the *2-adic perturbation* $v \to v^\sigma$, parametrized by $\sigma > 0$, and explore some of its formal properties. The 2-adic perturbation will serve as a building block for our perturbation scheme for multi-unit auctions, to be defined in Section 9.5.

For a positive integer j, we let $\tau(j)$ be the exponent of the largest power of 2 dividing j. The function τ appears in many contexts, and is known as the 2-*adic valuation function*, among other names. We illustrate the structure of τ by the plot in Figure 9.1. There are multiple equivalent definitions of τ: $\tau(j)$ is the number of trailing zeros in the binary representation of j, and equivalently $\tau(j) = t$ if and only if $j/2^t$ is an odd integer. For convenience, we set $\tau(0) = 0$. It is immediate that $\tau(j)$ can be evaluated in time polynomial in the length of the binary representation of j. Since we will need to evaluate τ only for integers between 1 and m, this is possible in time polynomial in $\log m$.

Given a non-decreasing valuation function $v : \{0, \ldots, m\} \to \mathbb{R}_+$ and $\sigma > 0$, we define the the perturbed valuation v^σ as follows:

$$v^\sigma(j) = v(j) + 2\sigma \cdot \tau(j) \tag{9.2}$$

The 2-adic perturbation can be viewed as one that gives a "bonus" to the valuation at each point j, proportional to the 2-adic valuation $\tau(j)$ of j. This bonus varies in a periodic and hierarchical pattern with the number j, as shown in the plot of the 2-adic valuation in Figure 9.1. The 2-adic perturbation has the effect of reducing the number of dominant points of a valuation, as seen in Figure 9.2.

Looking ahead to Section 9.5, our perturbation scheme for multi-unit auctions will in effect compose the 2-adic perturbation (with some carefully chosen value of σ to be defined) with the random singleton perturbation scheme of Chapter 8 (Algorithm 8.2). Because the random singleton perturbation scheme may change the number of dominant points of a valuation in general, we will not only bound the number of dominant points of a valuation v^σ perturbed by the 2-adic perturbation, but we will also bound the number of points that are even *near-dominant*. This would "leave room" for a variant of the RS perturbation scheme to act on v^σ, without substantially increasing the number of its dominant points.

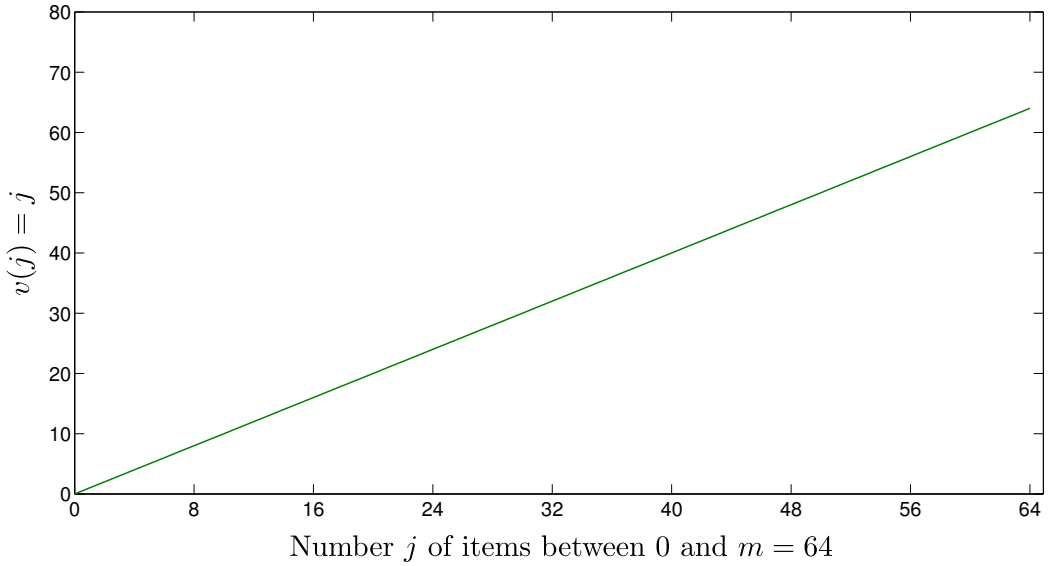
Given a function $u : [m] \to \mathbb{R}_+$, we say $k \in [m]$ is a *σ-dominant point of u* if $u(k) > u(\ell) - \sigma$ for all $\ell < k$. Observe that a point is dominant (Section 9.3) if and only if it is 0-dominant. We note that, if u is non-decreasing and $\sigma > 0$, then every point of u is σ-dominant, rendering the definition of approximate dominance uninteresting. The 2-adic perturbation is designed to effectively reduce the dimensionality of the valuation function from $[m]$ to a polynomial in $\log m$ and $\frac{v([m])}{\sigma}$, in the following strong sense:

Lemma 9.4.1. *Fix $\sigma > 0$ and a non-decreasing function $v : [m] \to [0, v_{\max}]$. The number of σ-dominant points of v^σ is at most $O(\log m \cdot \frac{v_{\max}}{\sigma})$. Moreover, given only value-oracle access to v, there is an algorithm that runs in $\mathrm{poly}(\log m, \frac{v_{\max}}{\sigma})$ time, and outputs a subset of $[m]$ guaranteed to include all σ-dominant points of v^σ.*

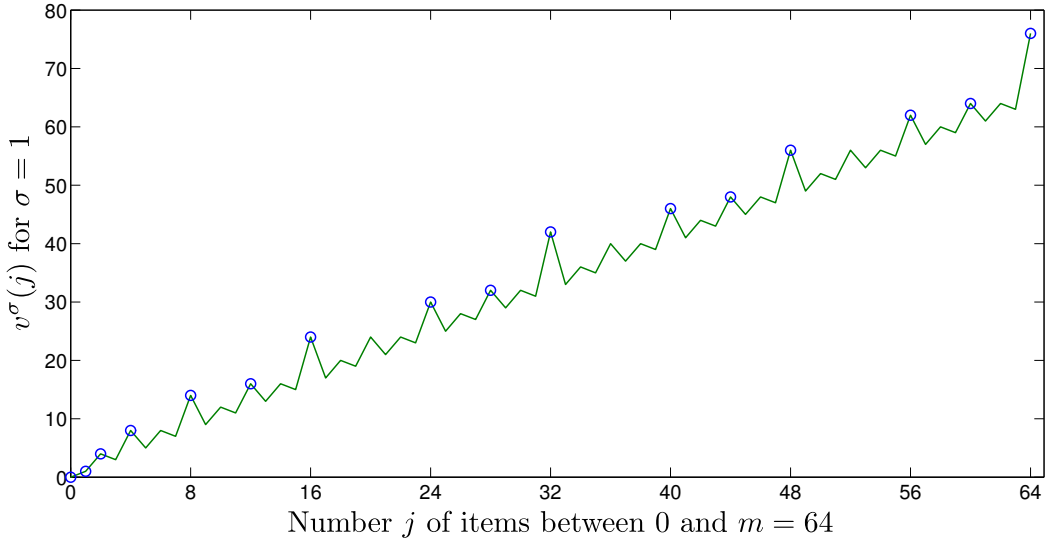
The proof of Lemma 9.4.1 builds on the following claim.

Claim 9.4.2. *Fix $\sigma > 0$ and non-decreasing function $v : [m] \to [0, v_{\max}]$. If k is a σ-dominant point of v^σ and $j < k$, then either $v(k) > v(j) + \sigma$ or $\tau(k) > \tau(j)$.*

Proof. We assume that $j < k$, $v(k) \leq v(j) + \sigma$, and $\tau(k) \leq \tau(j)$, and show that k is not a σ-dominant point of v^σ. Recall that $\tau(j)$ and $\tau(k)$ are the number of trailing



(a) Unperturbed valuation $v(j) = j$. All points are dominant.



(b) The 2-adic perturbation of v with $\sigma = 1$. Dominant points are circled.

Figure 9.2: The 2-adic perturbation.

zeros in the binary representation of j and k, respectively. Let $\ell = k - 2^{\tau(k)}$. Using the binary-representation interpretation of τ, it is easy to verify that ℓ is simply the result of removing the rightmost 1 from the binary representation of k, and therefore $\tau(\ell) \geq \tau(k) + 1$. Moreover, since the binary representation of j has at least $\tau(k)$ trailing zeros and $j < k$, it follows that $j \leq l$. To derive a contradiction, we will show that k does not σ-dominate ℓ in v^σ:

$$\begin{aligned} v^\sigma(\ell) &= v(\ell) + 2\sigma\tau(\ell) \\ &\geq v(\ell) + 2\sigma\tau(k) + 2\sigma \\ &\geq v(j) + 2\sigma\tau(k) + 2\sigma \\ &\geq v(k) - \sigma + 2\sigma\tau(k) + 2\sigma \\ &= v^\sigma(k) + \sigma \end{aligned}$$

□

Proof of Lemma 9.4.1. We will bound the number of σ-dominant points of v^σ by enumerating a list of at most $O(\log m \cdot \frac{v_{\max}}{\sigma})$ points that must include all σ-dominant points. First, we divide $\{0, \ldots, m\}$ into $O(\frac{v_{\max}}{\sigma})$ disjoint intervals, the sth of which is defined as follows:

$$I_s = \{j \in [m] : \sigma s < v(j) \leq \sigma(s+1)\}.$$

Claim 9.4.2 implies that, for each interval I_s and integer $t \in \{0, \ldots, \log m\}$, there is at most one σ-dominant point k of v^σ with both $k \in I_s$ and $\tau(k) = t$. This bounds the number of σ-dominant points of v^σ by $O(\log m \cdot \frac{v_{\max}}{\sigma})$, completing the proof of the first part of this Lemma.

Claim 9.4.2 implies an even stronger statement: if $k \in I_s$ is a σ-dominant point of v^σ and $\tau(k) = t$, then k must be the smallest integer j in I_s with $\tau(j) = t$. It is easy to verify that, given the end-points of a segment I_s and an integer t, we can compute in $O(\log m)$ time the smallest integer $k \in I_s$ that is divisible by 2^t, if any. Moreover, the delimiting points of the intervals $\{I_s\}_s$ can each be computed in $O(\log m)$ time using binary search. This completes the proof of the second part of the lemma. □

9.5 Combining the 2-adic and Random Singleton Perturbation Schemes

We now describe a perturbation scheme combining the random singleton (RS) perturbation scheme (Algorithm 8.2) with the 2-adic perturbation. We call this the *2-adic random singleton perturbation scheme*, and denote it by RS2. As we did for the RS scheme in Chapter 8, we describe RS2 first via its adjoint, depicted in Algorithm 9.1. Recall that each x in the feasible set Ω of multi-unit auctions (integer program (9.1)) is a binary vector indexed by $i \in [n]$ and $j \in [m]$, where x_{ij} denotes whether player i receives exactly j items. We assume that each $x \in \Omega$ is represented succinctly as a mapping from each player to a number of items. We observe that the probabilities in Step 2 of the adjoint RS2 scheme sum to 1, because a binary vector x that encodes a feasible solution for multi-unit auctions includes at most n non-zero entries, and moreover $\delta_{ij} + \delta'_{ij} \leq \epsilon/n$ for each i and j.

Parameter: $\epsilon > 0$.
Input: Numbers m of items and number n of players.
Definition: $\Omega \subseteq \{0,1\}^{[n]\times[m]}$ is the feasible region of integer program (9.1).
Input: $x \in \Omega$.
Output: $y \in \Omega$.

1: for each $i \in [n]$ and $j \in [m]$, draw δ_{ij} uniformly from the interval $[0, \frac{\epsilon}{4n \log m}]$, and let $\delta'_{ij} = \frac{\epsilon}{2n \log m}\tau(j)$ where τ is the 2-adic valuation defined in Section 9.4;
2: Output a random solution $y \in \Omega$ according to the following distribution:

- x with probability $1 - \epsilon$;
- for each player $i = 1, \ldots, n$, the "singleton" e_{im} (which assigns all items to i) with probability
$$\sum_{i,j} \frac{(\delta_{ij} + \delta'_{ij})x_{ij}}{n}; \tag{9.3}$$
- the all-zero allocation with the remaining probability.

Algorithm 9.1: Adjoint of the 2-adic random singleton (RS2) perturbation scheme.

We observe two main differences between the adjoint forms the RS and RS2 perturbation schemes (Algorithms 8.1 and 9.1, respectively). First, recall that the RS scheme gives an allocation x a "bonus", in the form of a random singleton allocation, with probability proportional to the number of non-zero entries of x. Polynomial dimension guarantees that the value of the random singleton bonus is at the right scale, up to a polynomial, so that the perturbation is diffuse enough to permit polynomial smoothed complexity. In exponential dimension, this is no longer the case since only an exponentially small fraction of singletons may be of sufficient value. Therefore, RS2 employs a random choice of n specific singletons — those allocating all items to a single player. The structure of multi-unit auctions implies that the expected value of this random bonus is at the right scale, up to a factor of n. The second and perhaps most important difference between RS and RS2 concerns the probability with which the bonus is awarded. Whereas in RS this probability depends only on the number of non-zero entries of x, in RS2 this probability depends additionally on the specific entries of x that are non-zero, using the 2-adic valuation to effectively place "weights" on various entries of x.

After conditioning on the random choices in Step 1 of the adjoint RS2 scheme (Algorithm 9.1), the expectation $\widetilde{x}$ of the distribution D_x over solutions $y \in \Omega$ defined by Step 2 can be expressed via the adjoint perturbation P^T given by

$$\widetilde{x} = P^T x = (1-\epsilon)x + \left(\sum_{i=1}^{n}\sum_{j=1}^{m}(\delta_{ij}+\delta'_{ij})x_{ij}\right)\left(\sum_{i=1}^{n}\frac{e_{im}}{n}\right)$$

Let δ and δ' denote the $n \times m$-vectors of δ_{ij}'s and δ'_{ij}'s, respectively. Since P^T can be written as $(1-\epsilon)I + \frac{1}{n}(\sum_{i=1}^{n} e_{im})(\delta+\delta')^T$, dualizing gives the following primal form of the RS2 perturbation matrix:

$$P = P(\delta) = (1-\epsilon)I + \frac{(\delta+\delta')(\sum_{i=1}^{n} e_{im})^T}{n} \tag{9.4}$$

This corresponds to the linear perturbation given by the map

$$v_i(j) \mapsto (1-\epsilon)v_i(j) + \frac{\delta_{ij} + \delta'_{ij}}{n} \sum_{i'=1}^{n} v_{i'}([m]) \tag{9.5}$$

for each player i and number of items j. We summarize the resulting (primal) form of the RS2 perturbation scheme in Algorithm 9.2.

Parameter: $\epsilon > 0$.
Input: Numbers m of items and number n of players.
Input: Valuation functions $v_1, \ldots, v_n : [m] \to \mathbb{R}_+$
Output: Perturbed valuation functions $\widehat{v}_1, \ldots, \widehat{v}_n : [m] \to \mathbb{R}_+$

1: For each $i \in [n]$ and $j \in [m]$, draw δ_{ij} uniformly from the interval $[0, \frac{\epsilon}{4n \log m}]$, and let $\delta'_{ij} = \frac{\epsilon}{2n \log m}\tau(j)$ where τ is the 2-adic valuation defined in Section 9.4;
2: Let

$$\widehat{v}_i(j) = (1-\epsilon)v_i(j) + \frac{\delta_{ij} + \delta'_{ij}}{n} \sum_{i'=1}^{n} v_{i'}([m])$$

for each player i and number of items j

Algorithm 9.2: The 2-adic random singleton (RS2) perturbation scheme.

9.6 The Truthful-in-Expectation FPTAS

In this section, we show that the RS2 perturbation scheme can be used in conjunction with the modified PB allocation rule (Section 9.3) to yield a truthful-in-expectation FPTAS for multi-unit auctions. We begin by proving an analogue of Lemma 8.3.1.

Lemma 9.6.1. *The RS2 perturbation scheme (Algorithm 9.2) for multi-unit auctions is feasible, linear, approximation-preserving, and tractable with deferred decisions.*

Proof. The proof of feasibility, linearity, and approximation-preservation is identical to that in Lemma 8.3.1. We now prove the scheme is tractable with deferred decisions (recall properties (I), (II), and (III) of Section 9.3). Property (I) is satisfied because the perturbation matrix P (Equation (9.4)) is uniquely defined by the mn independent

random variables δ_{ij}, each of which can be sampled in constant time. Property (II) is satisfied because the perturbed entry $(Pv)_{ij}$, as seen in Equation (9.5), can be evaluated in $O(n)$ time given oracle access to entries of the original valuations v, and given the value of δ_{ij}. For property (III), we recall that we described the RS2 perturbation explicitly via its adjoint. Specifically, Algorithm 9.1 on input x samples a distribution D_x with expectation $P^T x$. Moreover, it is clear that sampling of D_x reduces to evaluation of the probability $\sum_{ij}(\delta_{ij} + \delta'_{ij})x_{ij}/n$ (Equation (9.3)). This is possible in polynomial time because a feasible solution x for multi-unit auctions has at most n non-zero entries, δ_{ij} can be sampled in constant time, and evaluation of δ'_{ij} reduces to evaluating the 2-adic valuation $\tau(j)$ of j, which we argued in Section 9.4 is possible in time polynomial in $\log m$. □

We also show that instances of multi-unit auctions can perturbed by the RS2 perturbation scheme and then solved exactly, all in expected polynomial time.

Lemma 9.6.2. *Let $\epsilon > 0$ be a parameter. Let $v_1, \ldots, v_n : [m] \to \mathbb{R}_+$ denote an instance of multi-unit auctions, presented as n value oracles. Let $\delta_{ij} \in [0, \frac{\epsilon}{4n \log m}]$ for $i, j \in [n] \times [m]$, and assume δ is presented as an oracle indexed by i and j. There is an algorithm $\mathcal{A}$ that outputs an exact solution for perturbed instance Pv of multi-unit auctions, where $P = P(\delta)$ is the perturbation of the RS2 scheme with random coins δ as given in Equation (9.4), and runs in expected time polynomial in n, $\log m$, and $1/\epsilon$ when δ_{ij} are independent and uniformly distributed in $[0, \frac{\epsilon}{4n \log m}]$.*

Proof. Let $\sigma = \frac{\epsilon}{4n \log m} \cdot \frac{\sum_{i=1}^n v_i([m])}{n}$, and let $\eta_{ij} = \delta_{ij} \cdot \frac{\sum_{i=1}^n v_i([m])}{n}$. Plugging in the definition of δ' from Algorithm 9.2, the map $v \to Pv$ of the RS2 perturbation scheme, as given by Equation (9.5) for each δ, can immediately be re-written as follows:

$$\widehat{v}_i(j) = (1 - \epsilon)v_i(j) + 2\sigma\tau(j) + \eta_{ij}, \tag{9.6}$$

where η_{ij} are independently and uniformly distributed in $[0, \sigma]$. Equation (9.6) allows us to describe the RS2 perturbation scheme as the composition of two steps. For each player i, the RS2 scheme (a) scales player i's valuation v_i by $1 - \epsilon$, and applies the 2-adic perturbation (Section 9.4) with parameter σ to the scaled valuation, then (b) adds noise η_{ij}, independent and uniform in $[0, \sigma]$, to each entry j of the valuation.

We first show that, for a player i, there is a polynomial number of points $B_i \subseteq [m]$ independent of the "noise" vector η, such that all dominant points of $\widehat{v}_i$ lie in B_i, and moreover the list B_i can be computed in polynomial time. We let $\widetilde{v}_i(j) = (1-\epsilon)v_i(j) + 2\sigma\tau(j)$ be the result of applying only part (a) of the RS2 perturbation, and note that $\widehat{v}_i(j) = \widetilde{v}_i(j) + \eta_{ij}$. Observe that a dominant point of $\widehat{v}_i$ is a σ-dominant point of $\widetilde{v}_i$, because $0 \leq \eta_{ij} \leq \sigma$ for all i and j. Moreover, Lemma 9.4.1 implies that a list of σ-dominant points of $\widetilde{v}_i$ can be be computed in time polynomial in $\log m$ and $\frac{\widetilde{v}_i([m])}{\sigma}$, where the latter quantity is at most $\frac{4n^2 \log m}{\epsilon}$ by definition of σ.

For each player i, an optimal solution must assign i a number of items that is dominant for $\widehat{v}_i$. Therefore, after computing the sets B_i for each player i, we can re-write the problem of maximizing the perturbed welfare $\sum_i \widehat{v}_i(j)x_{ij}$ as a binary packing problem of polynomial dimension, with a variable x_{ij} for each player i and $j \in B_i$. Moreover, this reformulation is independent of the "noise" vector η. As a result, ours is a perturbed instance where each entry of the objective function is independent and has density $\frac{1}{\sigma} \leq \frac{4n^2 \log m}{\epsilon v_{max}}$, where $v_{max} = \max_{ij} v_i(j)$. Using proposition 8.2.1 and an FPTAS for multi-unit auctions, this perturbed instance can be solved in expected time polynomial in n, $\log m$, and $1/\epsilon$. This completes the proof. □

Combining the two Lemmas 9.6.1 and 9.6.2 with Theorem 9.3.1, we finally prove the existence of a truthful-in-expectation FPTAS for multi-unit auctions.

Theorem 9.6.3. *There is a truthful-in-expectation FPTAS for multi-unit auctions.*

Proof. Let $\mathcal{A}$ denote an algorithm for perturbing and solving an instance of multi-unit auctions, as described in Lemma 9.6.2. Let Ψ denote the RS2 perturbation scheme (Algorithm 9.2). Instantiate the modified PB allocation rule (recall form Section 9.3) for multi-unit auctions with the scheme Ψ and algorithm $\mathcal{A}$. Since Ψ is feasible, linear, approximation-preserving and tractable with deferred decisions (Lemma 9.6.1), Theorem 9.3.1 implies that this allocation rule is MIDR, has an approximation guarantee of $1-\epsilon$ in expectation (for an arbitrary supplied parameter ϵ), and has expected running time bounded by a polynomial in n and $\log m$ plus the runtime of Algorithm $\mathcal{A}$. Lemma 9.6.2 then bounds the expected runtime by a polynomial in n and $\log m$.

Finally, to complete the proof we note that Proposition 3.4.1 implies that truth-telling payments for this allocation rule can be computed with poly(n) overhead in runtime. □

Part IV

Mechanisms for Single-Parameter Problems

Chapter 10

Single-Parameter Scheduling Problems

10.1 Introduction

The previous chapters of this thesis were primarily concerned with mechanism design problems in which a player's preferences are described by multiple private parameters. Those problems are called *multi-parameter*, and their complexity frequently restricts the class of incentive-compatible mechanisms. As a result, the vast majority of mechanisms designed for multi-parameter problems fall in the VCG framework. Much more permissive is the class of *single-parameter* mechanism design problems, where each player's private valuation is described by a single real number. The class of truthful mechanisms for these problems is much better understood, and known to be more permissive.

In this chapter, we consider single-parameter problems in a scheduling context. We focus primarily on the problem of minimizing the makespan of parallel related machines, denoted in literature on scheduling by $Q||C_{\max}$. In this problem, a set of jobs with known sizes must be scheduled on a set of self-interested machines (the players). Each machine holds its speed privately, and the goal is find a schedule that minimizes the completion time of all jobs, known as the *makespan*. This single-parameter problem was considered the last serious candidate problem for a separation between the

power of polynomial-time truthful mechanisms and polynomial-time approximation algorithms in single-parameter mechanism design. Our result is positive: we present the first truthful-in-expectation randomized polynomial-time approximation scheme (PTAS) for $Q||C_{\max}$, improving upon the previously best known 2-approximation mechanism due to Archer and Tardos [3]. Our approximation guarantee is the best possible for all polynomial-time algorithms (assuming $P \neq NP$).

To obtain our result for $Q||C_{\max}$, we develop several algorithmic techniques for the design of incentive-compatible mechanisms for single-parameter scheduling problems. These techniques are flexible, and also yield truthful-in-expectation approximation schemes for objectives other than the makespan of the schedule.

10.1.1 Single Parameter Problems and Scheduling

In the problems considered in this chapter, each player's valuation function is negative. Therefore, for notational convenience, we flip the signs and define single-parameter problems when each player is equipped with a private *cost function* defined on the outcomes.

A mechanism design problem with m players is *single-parameter* if all outcomes $\omega \in \Omega$ are real m-vectors, and each player i's private cost function has the form $t_i(\omega) = c_i\omega_i$ for a private real number c_i. In the problems we consider, players correspond to m machines, and there are n jobs with known sizes $p_1, \ldots, p_n$. Each outcome $\omega \in \Omega$ corresponds to a *schedule*, an assignment of jobs to machines, and ω_i is the *work* (sum of job sizes) assigned to machine i. The cost function t_i has the form ω_i/s_i, where s_i denotes the (private) speed of player i's machine. In this case, we refer to player i's cost $t_i(\omega)$ also as the *load* on his machine, or alternatively the *completion time* of the machine.

In our primary application, our objective is to minimize the *makespan* $\max_i t_i(\omega)$ of the machines. Every vector s of private speeds induces an instance of an optimization problem Π, which in this case is the strongly NP-hard problem $Q||C_{\max}$. We note that, unlike other problems considered in this thesis, the objective in $Q||C_{\max}$ is not the utilitarian objective. Fortunately, single parameter problems admit a larger

class of truthful mechanisms than multi-parameter problems, enabling more positive results for non-standard objectives. We describe the space of truthful-in-expectation mechanisms for single-parameter problems next.

An algorithm for a single-parameter problem is *monotone* if increasing the value of a c_i (keeping other c_j's fixed) can only decrease the ith component of its solution in expectation. In our scheduling context, monotonicity means that slowing down one machine can only decrease the expected work assigned to it by the algorithm (for fixed speeds of the other machines). Prior work in mechanism design shows that an algorithm for a single-parameter problem is implementable — i.e., via suitable payments, it can be extended to a truthful-in-expectation mechanism — if and only if it is monotone. Conceptually, *polynomial-time single-parameter mechanism design is equivalent to polynomial-time monotone algorithm design.* Variants of this characterization were proved in different contexts, starting with the work of Mirrlees [66] and Spence [82], then Myerson [68], and finally made explicit for dominant-strategy truthfulness by Archer and Tardos [3]. For completeness, we review this characterization and present the form of the corresponding payment rule in Appendix A.3.

10.1.2 Results

Our main result is the first randomized monotone PTAS[1] for the $Q||C_{\max}$ problem. Using standard techniques for computing payments, we obtain a polynomial-time, truthful-in-expectation mechanism whose approximation guarantee is the best possible for polynomial-time algorithms (assuming $P \neq NP$). Prior to our work, the best polynomial-time and truthful-in-expectation mechanism was a 2-approximation due to Archer and Tardos [3].

The algorithmic techniques we develop for this result are flexible and easily yield additional truthful-in-expectation mechanisms for various single-parameter problems: a deterministic quasipolynomial-time approximation scheme (QPTAS) for $Q||C_{\max}$ (improving over [1]); a randomized PTAS and deterministic QPTAS for minimizing

[1] A *polynomial time approximation scheme* for a minimization problem Π is an algorithm that takes as input an instance of Π and a parameter $\epsilon > 0$, runs in time polynomial in the size of the instance, and returns a solution of cost at most a factor of $(1 + \epsilon)$ of the minimum possible.

the p-norm of loads on related machines; and a randomized PTAS for maximizing the minimum load on related machines (cf. [40]).

10.1.3 Techniques

We identify two key sources of non-monotonicity in classical approximation algorithms for $Q||C_{\max}$ and related problems, and develop a number of ideas to overcome them. Both the known PTASes for $Q||C_{\max}$ [39, 50] optimize over a compact but coarse representation of an allowable subset of schedules, represented as paths in a polynomial-size graph. This allowable subset fluctuates as a function of the machine speeds, so varying a machine speed causes unpredictable (and non-monotone) changes in algorithm behavior. Secondly, even when a machine speed perturbation leaves the allowable schedules invariant, attempting to optimize over their coarse representation inevitably yields only an approximate result. Approximation creates another opportunity for non-monotone behavior, with small perturbations in a machine speed potentially influencing the approximate solution chosen in an uncontrollable way.

To handle the first difficulty, we present a flexible approach for optimizing over a speed-independent set of schedules that yields monotone algorithms. Exact optimization over a fixed set is pervasive in algorithmic mechanism design, and is embodied by the maximal-in-distributional-range paradigm which is applied to several welfare maximization problems in this thesis. Since they specifically optimize social welfare, MIDR algorithms do not appear to be directly useful for problems such as $Q||C_{\max}$ that have a non-welfare objective. Nevertheless, we consider the natural analogue: minimizing expected makespan over a speed-independent set $\mathcal{R}$ of (possibly random) schedules. Unlike in the case of a welfare objective, this approach does not always lead to a monotone algorithm, and this is due to non-monotonicities that may arise in tie breaking among different schedules with the same makespan. We overcome this difficulty by exhibiting a flexible framework for "monotonicity-preserving tie breaking" that applies to ranges $\mathcal{R}$ that satisfy certain properties.

The second and more technically challenging task is to identify a set $\mathcal{R}$ of schedules that is rich enough to contain near-optimal solutions for all possible machine

speeds, yet structured enough to permit polynomial-time exact optimization. We use randomization twice to coax a job set into a form that enables polynomial-time optimization over the schedules. First, we artificially equalize the sizes of jobs that originally had similar sizes, randomly replacing them with the original job sizes at the end of the algorithm. We call this *job smoothing*. Second, we allow fractional schedules which we eventually convert to integral schedules via randomized rounding. Randomized rounding was also used by Archer and Tardos [2, 3], although our approximation target of $(1 + \epsilon)$ allows only the barest use of the technique: the jobs fractionally assigned to each machine must be dwarfed by those assigned fully.

At the end of the day, schedules in $\mathcal{R}$ fractionally assign the smoothed jobs. Such a fractional and smoothed schedule naturally corresponds, through application of randomized rounding and job smoothing, to a lottery over schedules of the original jobs. Therefore $\mathcal{R}$ is *distributional range*, in the sense of Chapter 3.

10.1.4 Related Work

The problem $Q||C_{\max}$ is the paradigmatic problem in single-parameter mechanism design (see e.g. [58]), and was considered a realistic candidate problem for a conditional separation between implementable and non-implementable polynomial-time approximation algorithms. The problem admits an (exponential-time) implementable optimal algorithm, but all classical polynomial-time approximation algorithms for it, such as the polynomial time approximation scheme (PTAS) designed by Hochbaum and Shmoys [50], are not monotone [3]. Archer and Tardos [3] devised a polynomial-time monotone randomized approximation algorithm that is 3-approximate. Archer [2] later modified the algorithm and analysis to improve the performance guarantee to 2.

We also mention prior related work for the other scheduling problems we consider. For the problem of minimizing a norm of machine loads, a non-monotone PTAS was given by Epstein and Sgall [39], but no nontrivial monotone algorithms were known prior to our work. For the problem of maximizing the minimum load, a non-monotone PTAS was given by Azar and Epstein [6], and a monotone PTAS for the special case of a constant number of machines was given by Epstein and van Stee [40].

We note that since the conference publication of the results in this chapter, our result for $Q||C_{\max}$ has been strengthened to a deterministic truthful PTAS by Christodoulo and Kovács [20].

10.2 A Monotone PTAS for Minimizing Makespan

10.2.1 Techniques for Monotone Algorithm Design

This section identifies a large class of monotone randomized algorithms for $Q||C_{\max}$, together with additional (strong) conditions that ensure an approximation ratio of $(1+\epsilon)$. Sections 10.2.2-10.2.4 design a polynomial-time algorithm that meets all of these requirements. We first formally state the monotonicity requirement.

Definition 10.2.1 (Monotone Algorithm)**.** *A randomized algorithm for the $Q||C_{\max}$ problem is* monotone *if the expected work assigned to a machine i — holding the set of jobs and the speeds of the other machines fixed — is always a non-decreasing function of the machine speed s_i.*

Monotone algorithms are important because they are precisely the algorithms that can be extended to truthful-in-expectation mechanisms (Appendix A.3). We use three main techniques to achieve monotonicity: job smoothing, fractional assignment, and monotonicity-preserving tie breaking, and we describe each next.

Job Smoothing

For an arbitrary group S of k jobs, we define the *smoothed version* of S as a set of k jobs, each of size equal to the average size $(\sum_{j \in S} p_j)/k$ of a job of S. Given a schedule that includes these smoothed jobs, a *random shuffle* replaces each of them with a distinct job from S, with each such bijection equally likely. The smoothed size of a job is the same as its expected size following this random instantiation.

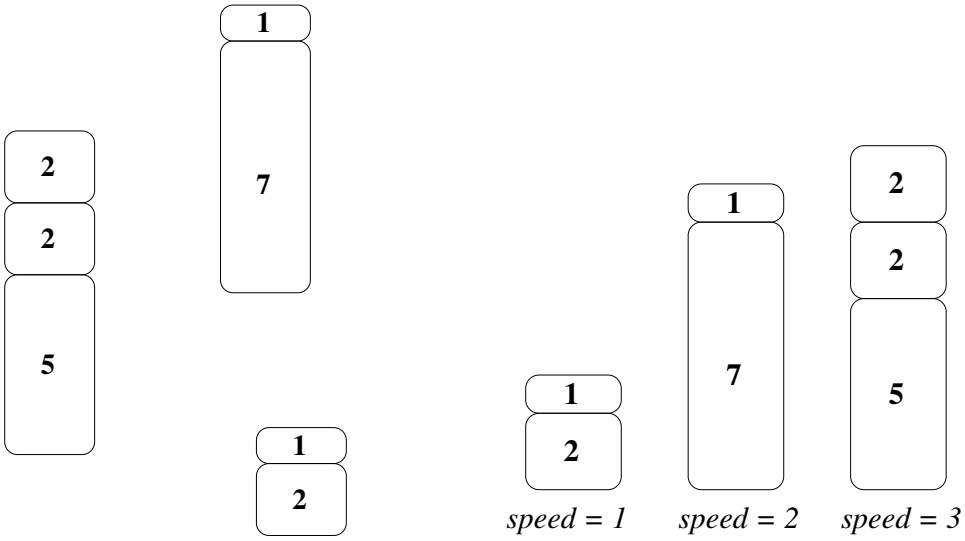
Fractional Assignment

As our second tool, we use the well-known technique of randomly rounding a fractional schedule. Precisely, a *fractional schedule* consists of a fractional assignment $\{y_{ij}\}_{i\in[m]}$ for each job j, where the y_{ij}'s are non-negative and sum to 1 for each j. The *makespan* of a fractional schedule is defined as the value of the maximum (fractional) load $(\sum_j p_j y_{ij})/s_i$ of a machine i. By *randomly rounding* a fractional schedule, we mean that each job j is independently assigned to a machine, according to the probability distribution $\{y_{ij}\}_{i\in[m]}$. The expected work on a machine following randomized rounding equals the work assigned to it in the fractional schedule.

Monotonicity-Preserving Tie Breaking

Finally, we require a technique to optimize over a set $\mathcal{R}$ of schedules without violating monotonicity. It was shown by Archer and Tardos [3] that finding a makespan-minimizing schedule in $\mathcal{R}$, and breaking ties using a total order on $\mathcal{R}$ fixed independently of machine speeds, results in a monotone algorithm. Unfortunately, implementing such a speed-independent tie-breaking rule appears to be computationally prohibitive. Instead, we present an approach for breaking ties among optimal schedules in $\mathcal{R}$ in a speed-dependent way that nevertheless preserves monotonicity.

Our monotonicity-preserving tie-breaking technique will apply only to sets $\mathcal{R}$ that are invariant under permuting the machines, in a sense we make clear next. Every fractional schedule of n jobs onto m machines induces an unordered *job partition* by ignoring the machine identities—a fractional partition of the n jobs into m classes, with each class corresponding to the job fractions assigned to a single machine. We sometimes call such a class S a *bundle*, and use $|S|$ to denote the corresponding amount of work (sum of fractional job sizes). Every job partition P naturally defines a set $\mathcal{R}(P)$ of $m!$ different fractional schedules, one for each bijection between bundles and machines. Given a set $\mathcal{X}$ of (fractional) job partitions, we let $\mathcal{R}(\mathcal{X}) = \cup_{P\in\mathcal{X}}\mathcal{R}(P)$, the set of all schedules defined by the partitions in $\mathcal{X}$. Our technique will apply to an arbitrary set $\mathcal{R}$ of fractional schedules of this form, defined in reference to a set of partitions $\mathcal{X}$.



(a) Job Partition (b) Induced Schedule

Figure 10.1: An unordered job partition and the induced schedule. Makespan of induced schedule is 4. Second machine is the bottleneck.

Before describing our monotone allocation rule, we need additional definitions. Given a job partition P, we use P_i to denote the class of a partition P with the ith-smallest amount of work (breaking ties arbitrarily). Given the speeds s of m machines, we single out the fractional schedule in which P_i is assigned to the ith slowest machine for each i, with ties between equal-speed machines broken in order of the machines' names, and call this the *schedule induced* by the given job partition and machine speeds. See Figure 10.1. A trivial exchange argument shows that this schedule minimizes makespan over all $m!$ arrangements of the partition, making it a natural choice. Given m machine speeds, the *makespan* of a job partition is the makespan of the fractional schedule it induces.

Let $\mathcal{X}$ be a set of (fractional) job partitions, each into m classes. We are now ready to describe our monotone allocation rule for choosing a makespan-minimizing schedule from $\mathcal{R}(\mathcal{X})$. Our allocation rule optimizes over the set $\mathcal{X}$, evaluating each partition via the makespan of the induced schedule, and breaks ties among optimal partitions according to a speed-independent total order $\prec$. Given the winning partition P from

the first step, the allocation rule outputs the schedule induced by P. Observe that this allocation rule breaks ties in a speed-dependent way between different schedules corresponding to the same partition.

Putting it all Together

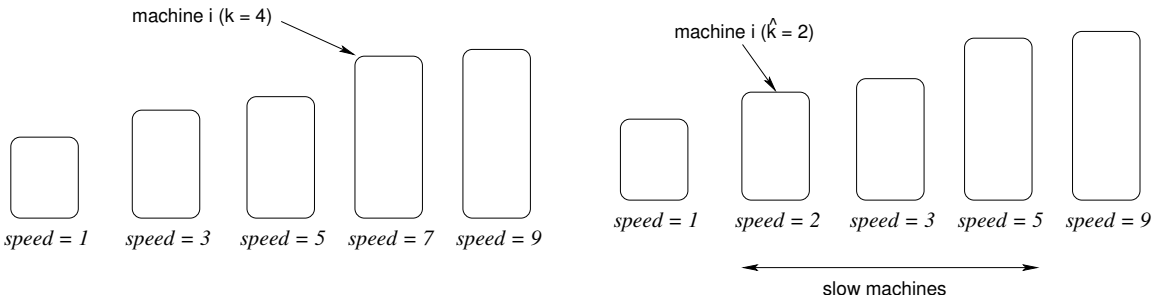
We combine our three main tools, job smoothing, fractional assignment, and the monotonicity-preserving tie-breaking rule, in a generic algorithm (Algorithm 10.1). The first step partitions the input jobs arbitrarily and applies job smoothing independently to each group. The second step fixes a set $\mathcal{X}$ of fractional job partitions and a total order $\prec$ on $\mathcal{X}$. The smoothing step and the definitions of $\mathcal{X}$ and $\prec$ are required to be independent of s. The third step optimizes over the permissible partitions $\mathcal{X}$ with respect to s. The final two steps transform the induced fractional schedule of the smoothed jobs into an integral schedule of the original jobs via randomized rounding and random shuffling. A short but slightly subtle proof shows that this algorithm is always monotone.

Input: n jobs with sizes $p_1, \ldots, p_n$ and m machines with speeds $s_1, \ldots, s_m$.

1: Group and smooth the jobs.
2: Define a set $\mathcal{X}$ of permissible fractional job partitions and a total ordering $\prec$ on $\mathcal{X}$.
3: Compute the partition in $\mathcal{X}$ with minimum makespan for s, breaking ties via $\prec$, and let σ_{frac} denote the induced schedule.
4: Transform σ_{frac} into an integral schedule σ_{smooth} of the smoothed jobs using randomized rounding.
5: Transform σ_{smooth} into an integral schedule of the original jobs using random shuffling.

Algorithm 10.1: A generic monotone algorithm. Only the third step depends on s.

Lemma 10.2.2 (Monotonicity of Generic Algorithm)**.** *For every speed-independent job grouping and choice of $(\mathcal{X}, \prec)$, Algorithm 10.1 is monotone.*



(a) Schedule σ (b) Schedule $\widehat{\sigma}$

Figure 10.2: Proof of Lemma 10.2.2.
The schedules induced by a job partition P for two speed vectors s and $\widehat{s}$ that differ only in the speed of machine i.

Proof. By the properties of randomized rounding and shuffling, the expected amount of work assigned to each machine equals the fractional work of the smoothed jobs assigned to it in the third step of the algorithm. Therefore, we only need to show that the fractional schedule of the smoothed jobs computed in the third step is monotone in the declared speeds.

Let $s = (s_i, s_{-i})$ and $\widehat{s} = (\widehat{s}_i, s_{-i})$ denote two speed vectors that differ only for machine i, with $s_i > \widehat{s}_i$, and let $P, \widehat{P} \in \mathcal{X}$ denote the corresponding optimal partitions. Let machine i be the kth slowest in s and the $\widehat{k}$th slowest in $\widehat{s}$, with $\widehat{k} \leq k$. Monotonicity demands that $|\widehat{P}_{\widehat{k}}| \leq |P_k|$.

Let σ and $\widehat{\sigma}$ denote the schedules induced by P for s and $\widehat{s}$, respectively. If both schedules have the same makespan, then P is also a $\prec$-minimum optimal schedule for $\widehat{s}$, so $\widehat{P} = P$ and $|\widehat{P}_{\widehat{k}}| = |P_{\widehat{k}}| \leq |P_k|$.

For the other case, call a machine *slow* if it is the ℓth slowest machine in $\widehat{s}$ and is strictly slower than the ℓth slowest machine in s. The parameter ℓ lies in $\{\widehat{k}, \ldots, k\}$ for each such machine; see Figure 10.2. If the makespan of $\widehat{\sigma}$ exceeds that of σ, then at least one slow machine, say the ℓth slowest in $\widehat{s}$, determines the makespan in $\widehat{\sigma}$ (since the load of each non-slow machines is no larger in $\widehat{\sigma}$ than in σ). The load on machine ℓ can only be less in the schedule induced by the optimal partition $\widehat{P}$ for $\widehat{s}$

— that is, $|\widehat{P}_\ell| \leq |P_\ell|$. Combining what we know completes the proof:

$$|\widehat{P}_{\widehat{k}}| \leq |\widehat{P}_\ell| \leq |P_\ell| \leq |P_k|,$$

where the first and third inequalities follow from the facts that $\widehat{k} \leq \ell \leq k$ and that bundle sizes are non-decreasing in an induced schedule. □

To control the approximation ratio of the generic algorithm (Algorithm 10.1), we impose three additional requirements — one for grouping, one for rounding, and one for the permissible job partitions.

Definition 10.2.3 (δ-Grouping)**.** *A partition of a set of jobs into groups is a* δ-grouping *if two jobs are in a common group only when their sizes are within a* $(1+\delta)$ *factor of each other.*

Definition 10.2.4 (δ-Integrality)**.** *A fractional job partition* P *is* δ-integral *if:*

(1) whenever a non-integral fraction of a job j *belongs to some class* P_i, $|P_i| \geq p_j/3\delta$*; and*

(2) every class of P *contains at most two fractional jobs.*

Definition 10.2.5 (δ-Good)**.** *A set* $\mathcal{X}$ *of permissible job partitions is* δ-good *if, for every speed vector* s, $\mathcal{X}$ *contains a partition with makespan at most* $(1+\delta)$ *times that of an optimal integral schedule.*

We have engineered these definitions in service of the next lemma.

Lemma 10.2.6 (Approximation Guarantee)**.** *Let* δ *be a sufficiently small positive constant. For every* $Q||C_{\max}$ *instance, every* δ*-grouping of the jobs, every* δ*-good set* $\mathcal{X}$ *of* δ*-integral partitions of the smoothed jobs, the schedule produced by the generic algorithm (Algorithm 10.1) has makespan* $1 + O(\delta)$ *times that of an optimal integral schedule, with probability 1.*

Proof. Fix a $Q||C_{\max}$ instance. Smoothing the jobs of this instance via a δ-grouping increases the makespan of an optimal schedule by at most a $(1 + \delta)$ factor. Algorithm 10.1 then computes an optimal permissible partition of the smoothed jobs;

since $\mathcal{X}$ is δ-good, the makespan of the induced (fractional) schedule is at most $(1+\delta)$ times that of an optimal integral schedule of the smoothed jobs, and at most $(1 + O(\delta))$ times that of an optimal integral schedule of the original jobs. By Definition 10.2.4, randomly rounding this δ-integral schedule increases the makespan by at most a $(1 + 6\delta)$ factor. By Definition 10.2.3, the random shuffling step increases the makespan further by at most a $1 + \delta$ factor. The generic algorithm thus terminates, with probability 1, with a $(1 + O(\delta))$-approximate solution to the original $Q||C_{\max}$ instance. □

10.2.2 Permissible Partitions

This section identifies a δ-grouping and a δ-good set $\mathcal{X}$ of δ-integral job partitions for use in the generic algorithm (Algorithm 10.1). Consider n jobs, a parameter m, and a positive constant δ. We can assume that $1/\delta$ is a sufficiently large power of 2. We begin with our δ-grouping procedure, which leads to what we call *bucket smoothing*. Group together all jobs that share the same values of two parameters: the largest W that is a power of 2 with $p_j > \delta W$ (call it W^*); and the unique i such that the job size belongs to the *ith W^*-bucket*, defined as the interval $(\delta W^*(1+(i-1)\delta), \delta W^*(1+i\delta)]$. This procedure only groups together jobs with sizes that differ by at most a $1 + \delta$ factor, so it is a δ-grouping. By design, smoothing enforces the following property: if the smoothed jobs j, k lie in a common W-bucket, where W is a power of 2 satisfying $p_j, p_k \in (\delta W, W]$ — possibly smaller than W^* above — then the jobs have the same size and are thus interchangeable.[2]

Bucket smoothing enables a succinct summary of the sizes of a set of jobs, described next. A *magnitude* is either 0 or a power of 2, and is used to determine the resolution at which we monitor job sizes. If W is a magnitude, we call a job *W-small* if its size is at most δW. If W is a magnitude that is at least the full size of each job in a collection of (possibly fractional) jobs S, then the *W-configuration of S* is a vector in which the first component (indexed by 0) denotes the total (fractional) work of the *W-small jobs* of S, divided by δW; and each of the other $\approx \frac{1}{\delta^2}$ components i counts

[2]Other grouping schemes, such as buckets of the form $(\delta W^*(1+\delta)^{i-1}, \delta W^*(1+\delta)^i]$, work equally well.

the number of jobs of S whose full size lies in the ith W-bucket. Bucket-smoothing ensures that all (non-W-small) jobs in the same W-bucket have equal size.

We now build up the defining properties of the job partitions that we include in our set $\mathcal{X}$.

Definition 10.2.7 (Legal Magnitudes)**.** *Let P denote a fractional job partition of bucket-smoothed jobs, with P_i denoting the ith smallest class. An m-vector w of magnitudes is* legal for P *if the following properties hold for each i:*

(P1) *w_i is a magnitude (0 or a power of 2) that is at least the full size of every job in P_i;*

(P2) *w_i is at least $1/\delta$ times the full size of every job that is fractionally assigned in P_i;*

(P3) $|P_i| \in [\frac{1}{3}w_i, \frac{7}{6}w_i]$.

Property (P2) of Definition 10.2.7 ensures that only w_i-small jobs can be fractional in P_i. Property (P3) ensures that there are no more than two legal values of w_i for a given P_i. Since the $|P_i|$'s are non-decreasing, legal w_i's must be *almost increasing* in the sense that $w_k \geq w_i/2$ whenever $k > i$.

If a partition P admits some vector w of legal magnitudes, and additionally each P_i contains at most two fractional jobs, then properties (P2) and (P3) together imply that P is δ-integral in the sense of Definition 10.2.4. The set of all such partitions is 0-good — for example, it includes all integral partitions — but appears far too rich to optimize over efficiently. This motivates our final properties, which impose just enough additional structure on the allowable job partitions to enable polynomial-time optimization without destroying δ-goodness.

Suppose w is legal for P. Place P_i in the *w_i-block* if $w_k \geq w_i$ for all $k > i$, and in the *$(w_i/2)$-block* if there is a $k > i$ with $w_k = w_i/2$. Note that if P_i is in the W-block, then $w_i \in \{W, 2W\}$. Since the w_i's are almost increasing, each block is a contiguous subset of the P_i's; see Figure 10.3. The final class P_i of a W-block (necessarily with $w_i = W$) is the *block endpoint*. The largest class P_m is always a block endpoint.

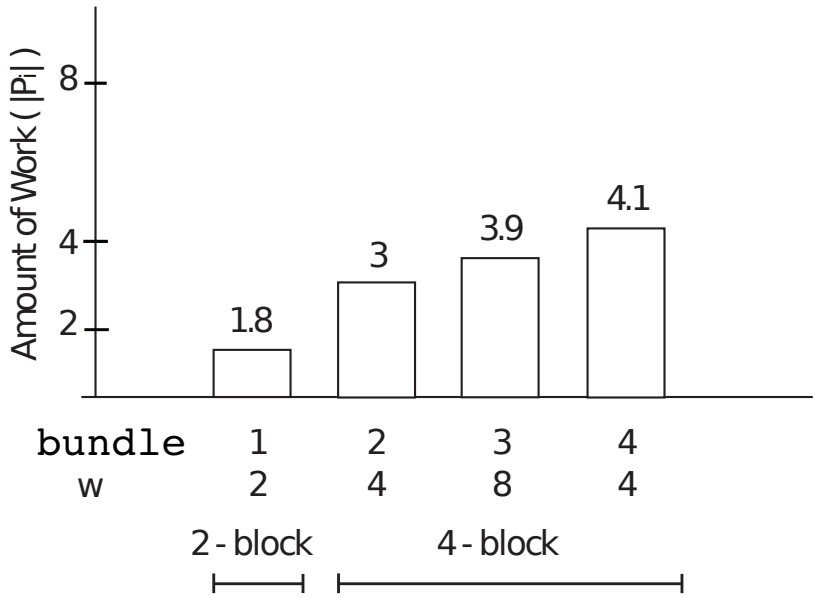


Figure 10.3: A job partition, legal magnitudes, and blocks.
This partition has 4 classes, and we depict legal magnitudes w for it, and the corresponding 2- and 4-blocks.

Definition 10.2.8 (Permissible Partitions)**.** *A fractional job partition P is* permissible *if it is δ-integral and there are legal magnitudes w such that:*

(P4) for every non-block endpoint P_i, the induced w_i-configuration of P_i is integral — i.e., the total (fractional) size of w_i-small jobs of P_i is a multiple of δw_i; and

(P5) for every block endpoint P_i other than P_m, the induced w_i-configuration of S_i is integral, where $S_i = \cup_{k\,:\,w_k \le w_i} P_k$.

Thus most classes of a permissible partition have integral configurations, and the cumulative configurations at certain milestones (the block endpoints) are also integral. These properties are essential for the existence of a polynomial-size representation of permissible partitions.

We take $\mathcal{X}$ to be the set of all permissible partitions, and order these partitions lexicographically by the work vector $(|P_1|, \ldots, |P_m|)$. (Ties between different partitions with the same work vector can be broken arbitrarily.) Neither the set $\mathcal{X}$ nor

this ordering $\prec$ depend on the machine speeds. The next two sections give proofs of the following technical but important lemmas.

Lemma 10.2.9 (δ-Goodness of Permissible Partitions)**.** *For every positive integer m, sufficiently small $\delta > 0$, and set of bucket-smoothed jobs, the corresponding set of permissible partitions is $O(\delta)$-good.*

Lemma 10.2.10 (Optimizing over Permissible Partitions)**.** *For every constant $\delta > 0$, the problem of computing the permissible partition of a bucket-smoothed $Q||C_{\max}$ instance with minimum makespan, breaking ties via $\prec$, can be solved in polynomial time.*

Since permissible partitions are δ-integral, Lemmas 10.2.2, 10.2.6, 10.2.9, and 10.2.10 imply our main result.

Theorem 10.2.11. *There is a randomized monotone PTAS for $Q||C_{\max}$.*

Applying the characterization of of implementable single-parameter allocation rules (Appendix A.3), as well as standard techniques for efficiently computing suitable payments (see Section 10.2.5), Theorem 10.2.11 yields a polynomial-time, $(1+\epsilon)$-approximate, truthful in expectation mechanism for $Q||C_{\max}$.

10.2.3 Proof of Lemma 10.2.9

Fix $\delta > 0$, which we can assume is at most a sufficiently small constant, an instance of $Q||C_{\max}$ with bucket-smoothed jobs J and speed vector s, and an optimal schedule σ^*. Rename the machines so that $s_1 \leq s_2 \leq \cdots \leq s_m$. We extract from σ^* a permissible partition with makespan (with respect to s) at most $1 + O(\delta)$ times that of σ^*.

Let W_{max} denote the smallest power of 2 that upper bounds the work of every machine in σ^*. We first create a *reserve* $R \subseteq J$ for subsequent "rounding up" of fractional configurations. Assume without loss that the smallest job size is 1. For $W = 1, 2, 4, \ldots, W_{max}$ in turn, greedily add W-small jobs to R until the total size of R is at least $3\delta W$ (and at most $4\delta W$), or until there are no such jobs to add.

If $|R| < 3\delta W_{max}$ at termination, then we can finish easily: we can transform σ^* into a schedule that induces a $(1 + 6\delta)$-approximate (integral) permissible partition by re-assigning the jobs of R to the machine with the most work, so that only the largest bundle contains small jobs. For the rest of the proof, we assume that $|R| \geq 3\delta W_{max}$.

The high-level proof plan is to begin with a set of legal weights (Definition 10.2.7), then enforce properties (P4) and (P5) of Definition 10.2.8, and finally restore δ-integrality; each step preserves the properties already established while increasing the makespan by a $1 + O(\delta)$ factor.

Delete from σ^* all jobs of R and permute the bundles so that work is non-decreasing in machine speed. Let $S_1, \ldots, S_m$ denote the corresponding bundles, an ordered partition of $J \setminus R$ indexed by machine name. For each i, define w_i as the unique power of 2 with $w_i/2 < |S_i| \leq w_i$; these are legal for the job partition induced by the schedule. Also, $w_m = \max_i w_i$ since $|S_m| = \max_i |S_i|$, and $w_m \geq W_{max}/2$ since $|R| \leq 4\delta W_{max}$ (for δ sufficiently small). We repeatedly transform the schedule in what follows; by definition, the w_i's remain fixed at their initial values throughout the process.

Call machine i *non-integral* if $i \neq m$ and if the w_i-configuration of its current (possibly fractional) bundle S_i is not integral — that is, the total (fractional) work created by the w_i-small jobs of S_i is not a multiple of δw_i. While there are two non-integral machines i and i', say with $w_i \leq w_{i'}$, we move w_i-small jobs from the former to the latter (where they are also $w_{i'}$-small), allowing fractional assignments, until one of the two machines becomes integral. This process terminates with at most one non-integral machine, say i. We conclude by moving w_i-small jobs from i to machine m — since $w_m = \max_i w_i$, they are also w_m-small — until the former becomes integral. This procedure terminates with a (fractional) schedule $(T_1, \ldots, T_m)$. Note that we *cannot* assume that $|T_1| \leq |T_2| \leq \cdots \leq |T_m|$. Nevertheless, this schedule induces a job partition meeting property (P4) of Definition 10.2.8.[3] Since it alters the amount of work assigned to each machine i by less than δw_i and only reschedules small jobs,

[3]Strictly speaking, this holds provided $|T_m| > \max_{i<m} |T_i|$; as we'll see, the mth bundle will satisfy this property by the end of the proof.

w remains legal for the job partition induced by the T_i's (for δ sufficiently small) and the makespan remains $(1 + O(\delta))$-approximate.

We dip into our reserve R to establish property (P5). We call a machine a *potential endpoint* if it is carrying a bundle that would be a block endpoint in the job partition induced by the current schedule and w. Precisely, machine i is a potential endpoint of the current schedule $T_1, \ldots, T_m$ if $w_k > w_i$ whenever $|T_k| > |T_i|$ and whenever $|T_k| = |T_i|$ and $k > i$. There is at most one potential endpoint per magnitude. A potential endpoint i is *non-integral* if $w_i < w_m$ and the w_i-configuration of $\cup_{k\,:\,w_k \leq w_i} T_k$ is not integral. While there is a non-integral potential endpoint, we pick the one (i, say) with smallest W-value, and move w_i-small jobs from R to machine i, again permitting fractional assignments, until it becomes integral. Adding these jobs cannot create new potential endpoints and strictly decreases the number of non-integral potential endpoints. At termination, the job partition induced by the final schedule $(U_1, \ldots, U_m)$ and magnitudes w satisfies property (P5). Every machine to which we added jobs is a block endpoint of this partition, so the procedure does not violate (P4). Less than δw_i work is added to a non-integral potential endpoint i, so w remains legal and the makespan is increased by only a $1 + O(\delta)$ factor. Also, R always contains enough jobs to implement each iteration: non-integrality of a potential endpoint i implies that not all w_i-small jobs are in R, so R began with at least $3\delta w_i$ units of w_i-small jobs; since W-values at least double each iteration, at most δw_i of these units were removed in prior iterations, leaving more than the requisite δw_i units available.

Once no non-integral potential endpoints remain, obtain the schedule $\widehat{\sigma}$ by assigning all remaining jobs of R — of total size between $2\delta W_{max}$ and $4\delta W_{max}$ — to machine m; this destroys neither the legality of w nor the $1+O(\delta)$ approximation factor. Since $|S_m| = \max_i |S_i|$ and both job re-assignment procedures add at most δw_i work to i without removing jobs from m, $|U_m| > \max_i |U_i| - 2\delta W_{max}$. Thus machine m has the most work in $\widehat{\sigma}$. The induced job partition, together with the legal magnitudes w, satisfies (P4) and (P5) of Definition 10.2.8.

To restore δ-integrality, remove the w-small jobs $V \subseteq J$ from $\widehat{\sigma}$, sort them in order of non-decreasing size, and re-assign them to machines in order of non-decreasing w_i so that the work assigned to each machine is the same as in $\widehat{\sigma}$ (using fractional

assignments only when needed). Call this final schedule σ. The makespan is obviously unchanged. One easily checks that all jobs of V remain small for their assigned machine(s) in σ, so legality of w and properties (P4), (P5) are preserved. Finally, a job of V is fractionally assigned in σ only if it is the final small job re-assigned to one machine and the first to another. Since every machine has at most two (small) fractionally assigned jobs in σ, the schedule induces a permissible partition.

10.2.4 Proof of Lemma 10.2.10

Input: n jobs with sizes $p_1, \ldots, p_n$ and m machines with speeds $s_1, \ldots, s_m$.

1: Construct a directed layered network with $m+2$ layers and a polynomial number of vertices in each layer. Layers 0 and $m+1$ contain only an origin o and a destination d, respectively.

2: Define edges between layers and associated edge lengths x so that every o-d path whose sequence of edge lengths has the form $0 \le x_1 \le x_2 \le \cdots \le x_m$ corresponds to a permissible partition P with $|P_i| = x_i$ for every i, and conversely.

3: Compute the o-d path Q^* that minimizes $\max_{i=1}^{m} x_i/s_i$ subject to having a sequence of edge lengths of the form $0 \le x_1 \le x_2 \le \cdots \le x_m$. Break ties lexicographically by the vector $(x_1, x_2, \ldots, x_m)$.

4: Return the permissible partition that corresponds to Q^*.

Algorithm 10.2: Optimizing over permissible partitions (Lemma 10.2.10).

This section shows that the problem of optimizing over permissible partitions can be solved exactly, with the requisite tie-breaking, in polynomial time. Algorithm 10.2 gives a high-level description, and the details follow. We first describe the layered shortest-path network, including motivation for its ingredients, and then give the precise correspondence between permissible partitions and certain paths in this network. Lemma 10.2.10 then follows easily.

Fix $\delta > 0$, m, and a set J of bucket-smoothed jobs. The graph G has $m+2$ layers; the first (0) and last $(m+1)$ contain only the origin o and destination d, respectively. For $i \in \{1, 2, \ldots, m\}$, the ith layer will consist of a polynomial number of vertices, each endowed with six labels. An edge from layer i to $i+1$ is meant to dictate the

ith-smallest bundle of a permissible partition and the corresponding magnitude w_i, as well as the value W of the W-block to which the $(i+1)$th bundle belongs (for $i+1 \leq m$). Our vertex labels will be rich enough so that the intentions of an edge can be inferred uniquely from the labels of its endpoints.

Precisely, every vertex in a layer $i \in \{1, 2, \ldots, m\}$ is labeled with two magnitudes W_1 and W_2. Each is required to be either 0 or in a polynomial-size set of powers of 2, ranging from the smallest power of two that upper bounds $p_{\min}$ to the smallest one that upper bounds $np_{\max}$, where $p_{\min}$ and $p_{\max}$ denote the smallest and largest job sizes. We also insist that $W_2 \geq 2W_1$. Choices of W_1, W_2 that meet these constraints are called *valid*. These labels are meant to indicate that the ith bundle belongs to the W_2-block — and thus its magnitude will be either W_2 or $2W_2$ — while the previous distinct block is the W_1-block.

The other four vertex labels A_1, B_1, A_2, B_2 summarize the sizes of the jobs assigned to the first $i-1$ bundles. Each is constrained to be an integral W-configuration for some magnitude W; there are only polynomially many ($n^{O(1/\delta^2)}$) such configurations. Configuration A_1 is meant to be the W_1-configuration of the set of jobs assigned to previous bundles $k < i$ with $w_k \leq W_1$; B_1 the $2W_1$-configuration of jobs in previous bundles $k < i$ in the W_1-block with $w_k = 2W_1$; A_2 and B_2 the W_2- and $2W_2$-configurations of jobs in previous bundles $k < i$ in the (current) W_2-block with $w_k = W_2$ and $w_k = 2W_2$, respectively. Four distinct labels are required to faithfully capture properties (P4) and (P5) of permissible partitions as integrality constraints on configurations.

Our intents for the labels A_1, B_1, A_2, B_2 suggest additional constraints. To explain them, recall that a W-configuration has a component (indexed by 0) indicating the total (possibly fractional) size of W-small jobs, divided by δW; and $\approx 1/\delta^2$ components that count the number of jobs in each W-bucket. We call a W-configuration C *realizable* if the total size of the W-small jobs of J is at least $C_0 \cdot \delta W$, and for each $i > 0$, at least C_i jobs of J belong to the ith W-bucket. Next, note that a W-configuration can be uniquely rewritten as a W'-configuration at a coarser resolution $W' \geq W$, with jobs moving to lower-indexed buckets, and some non-W-small jobs becoming W'-small. Thus two configurations with different magnitudes can be

sensibly added to produce one at the larger magnitude (though the sum of two integral configurations can have a fractional first component). Finally, we call the parameters $W_1, W_2, A_1, B_1, A_2, B_2$ *valid* if W_1, W_2 are valid, and every subset of $\{A_1, B_1, A_2, B_2\}$ sums to a realizable configuration (at the appropriate magnitude W_1, $2W_1$, W_2, or $2W_2$). Every layer $i \in \{1, 2, \ldots, m\}$ of G has one vertex for every possible set of valid parameters.

Next we describe the edge set of G, beginning with the edges from layer i to $i+1$ for $i \in \{1, 2, \ldots, m-1\}$. Let $(W_1, W_2, A_1, B_1, A_2, B_2)$ be the (valid) parameters of a vertex u in layer i. Let N_u denote the vertices v of layer $i+1$ that meet one of the following three conditions:

(A) all of v's parameters match those of u except for its fifth parameter, which is some configuration that is (component-wise) at least A_2;

(B) all of v's parameters match those of u except for its sixth parameter, which is some configuration that is (component-wise) at least B_2;

(C) v's parameters are $(W_2, W_3, D, B_2, 0, 0)$, where D is some integral W_2- configuration that is component-wise at least the (possibly fractional) W_2-configuration $A_1 + B_1 + A_2$.

These three cases are meant to correspond to the following scenarios: (A) bundle $i+1$ also belongs to the (current) W_2-block and $w_i = W_2$; (B) bundle $i+1$ also belongs to the (current) W_2-block but $w_i = 2W_2$; and (C) bundle $i+1$ belongs to the W_3-block for some $W_3 > W_2$. In all three cases, we can extract from the labels of u, v a proposed magnitude w_{uv} for the ith bundle — W_2 in (A) and (C), $2W_2$ in (B). We can also infer a corresponding w_{uv}-configuration, which we can interpret as a proposed ith bundle: in (A), the increase in the fifth parameter; in (B), the increase in the sixth parameter; and $D - A_1 - B_1 - A_2$ in (C). For $v \in N_u$, let x_{uv} denote the amount of work represented by the corresponding w_{uv}-configuration C. Because the jobs J are bucket-smoothed, x_{uv} is uniquely defined as $C_0 \cdot \delta w_{uv}$ plus $\sum_{h>0} C_h \cdot z_h$, where z_h denotes the common size of every job of J that lies in the hth w_{uv}-bucket. Eying constraint (P3) of Definition 10.2.7, we connect vertex u to

every $v \in N_u$ for which $x_{uv} \in [\frac{1}{3}w_{uv}, \frac{7}{6}w_{uv}]$. We assign each such edge (u, v) a length of x_{uv}. We classify such edges as *type A*, *type B*, or *type C* according to the condition met by its endpoints' labels.

The edges incident to o and d are defined similarly. The origin is connected to all vertices v of layer 1 that possess a label in which all parameters but the second are zero; such an edge effectively determines the value of W for the first W-block, but does not determine any bundles. These edges are all assigned a length of zero and have no type. Finally, consider a node v of layer m with valid parameters $(W_1, W_2, A_1, B_1, A_2, B_2)$. We adopt W_2 as the proposed magnitude for the mth bundle. We connect v to d in G if and only if there is a realizable W_2-configuration C such that $A_1 + B_1 + A_2 + B_2 + C$ is the $2W_2$-configuration of the full set J of jobs, and the corresponding amount of work x_{vd} of C lies in $[\frac{1}{3}W_2, \frac{7}{6}W_2]$. (There can be more than one such configuration C, but all solutions represent the same amount of work.) Each such edge (v, d) is assigned a length of x_{vd} and is classified as a type C edge. This construction of the network G can be performed in polynomial time.

We now verify that our construction represents permissible partitions.

Lemma 10.2.12. *Let G denote the network corresponding to a bucket-smoothed instance of $Q||C_{\max}$ and a constant $\delta > 0$.*

(a) For every permissible partition P, there is an o-d path of G whose sequence of edge lengths is $0, |P_1|, |P_2|, \ldots, |P_m|$.

(b) Given an o-d path of G whose sequence of edge lengths is $0 \leq x_1 \leq x_2 \leq \cdots \leq x_m$, a permissible partition P with $|P_i| = x_i$ for every i can be constructed in polynomial time.

Proof. Consider a permissible partition P and corresponding legal weights w. For each $i \in \{1, \ldots, m\}$, P and w naturally induce a vertex $v_i = (W_1, W_2, A_1, B_1, A_2, B_2)$ of layer i of G: W_1, W_2 are defined so that P_i belongs to the W_2-block of P and the previous distinct block is the W_1-block (or $W_1 = 0$ if no such block exists); and A_1, B_1, A_2, B_2 are derived from P according to their intended meanings, discussed above. Since P satisfies properties (P4) and (P5) of Definition 10.2.8, all four configurations are integral. By construction and properties (P1)–(P3) of Definition 10.2.8,

$W_1, W_2, A_1, B_1, A_2, B_2$ are valid parameters, corresponding to some vertex v_i of layer i of G. The edge (o, v_1) is clearly present in G. Our definition of edge lengths in G ensures that $x_{v_i,v_{i+1}}$ equals the work $|P_i|$, so property (P3) implies that the edges (v_1, v_2), (v_2, v_3), $\ldots$, (v_m, d) are present in G. The sequence of edge lengths along this path is precisely $0, |P_1|, |P_2|, \ldots, |P_m|$.

Conversely, consider an o-d path of G with intermediate vertices $v_1, \ldots, v_m$ and a non-decreasing sequence of edge lengths. As outlined above, the edges of this path suggest magnitudes w and, for each i, a corresponding w_i-configuration C^i. (Recall that C^m can be inferred from $C^1, \ldots, C^{m-1}$ and the set J of all jobs.) For example, if the label of v_i is $(W_1, W_2, A_1, B_1, A_2, B_2)$, (v_i, v_{i+1}) is a type-A edge, and the fifth parameter of v_{i+1}'s label is A_2', then we define $w_i = W_2$ and $C^i = A_2' - A_2$. The components of C^i other than the first indicate how many jobs from the different w_i-buckets should be (integrally) assigned to the ith bundle, while the first component of C^i describes the total fractional size of w_i-small jobs that should be assigned to this bundle. Our realizability constraints ensure that these configurations can be translated into a job partition $P_1, \ldots, P_m$ in the obvious way, with the final small job assignments performed as in the last step of Section 10.2.3 — in non-decreasing order of magnitude and of job size, resorting to fractional assignments only when needed. This translation can be performed in polynomial time, ensures that $|P_i| = x_{v_i v_{i+1}}$ for every i, and enforces δ-integrality. The produced partition P clearly satisfies properties (P1) and (P2) with respect to magnitudes w. The definition of the edge set of G ensures that P and w satisfy (P3). For property (P4), observe that every fractional configuration C^i results from a type C edge (v_i, v_{i+1}), and the corresponding bundle P_i is necessarily a block endpoint of P with respect to w. To complete the proof, note that every block endpoint P_i arises from some type C edge (v_i, v_{i+1}), and property (P5) then follows immediately from the integrality of the third parameter of v_{i+1} (representing the jobs assigned to bundles $k \leq i$ with $w_k \leq w_i$). □

We now complete the proof of Lemma 10.2.10.

Proof of Lemma 10.2.10. Consider a bucket-smoothed $Q||C_{\max}$ instance and a constant $\delta > 0$. Rename machines so that $s_1 \leq s_2 \leq \cdots \leq s_m$. Form the (speed-independent) network representation G of permissible partitions described above, and assign a cost of x_{uv}/s_i to every edge (u, v) traveling from layer i to layer $i+1$. By Lemma 10.2.12, computing the permissible partition with minimum makespan for s in polynomial-time reduces to computing the o-d path of G that has a non-decreasing sequence of x-values and minimizes the bottleneck edge cost (breaking ties among optimal solutions lexicographically according to the vector of x-values).

We claim that the latter problem can be solved in polynomial time. First, ignoring the tie-breaking requirement, we can solve the problem either directly using dynamic programming, or via Dijkstra's algorithm after a simple graph transformation that eliminates o-d paths that do not have a non-decreasing sequence of x-values.

To implement the desired tie-breaking, we solve this problem repeatedly. Initially the first layer is "active". In the first iteration we compute a non-decreasing path with optimal bottleneck edge length M^* and some value a for x_1. In the second iteration, we delete all edges e from layer 1 to layer 2 with $x_e \geq a$, and recompute a non-decreasing minimum-bottleneck path. If the new optimal path has bottleneck edge length larger than M^*, then every optimal path in the original graph satisfies $x_1 \geq a$. Otherwise, we obtain a new path that is optimal in the original graph and has an x_1-value b that is smaller than a. In the former case, we return the discarded edges with $x_1 = a$ back to the graph, deactivate the first layer, and activate the next layer. In the latter case, we discard edges e from layer 1 to layer 2 with $x_e \geq b$ and repeat.

Inductively, the procedure above maintains the following invariant, where i denotes the currently active layer: in the current network, there is at least one non-decreasing path with bottleneck edge length M^*; and every such path in the current network minimizes lexicographically the vector $(x_1, x_2, \ldots, x_{i-1})$ over all such paths in the original network G. Since every iteration either deletes edges from the currently active layer or makes a later layer active, this procedure terminates in polynomial time. By the invariant, it terminates with the non-decreasing minimum bottleneck path of G, with ties broken lexicographically according to the vector of x-values. □

10.2.5 Computing Payments

To extend our randomized monotone PTAS for $Q||C_{\max}$ to a truthful-in-expectation mechanism, we compute suitable payments by integrating the "work curve" of each machine (depicted in Figure A.1) as described in Appendix A.3 . For a given $Q||C_{\max}$ instance with machine speeds s, this computation boils down to determining, for every machine i and alternative speed report s'_i, the expected amount of work that would have been assigned to machine i had it reported s'_i instead of s_i.

We accomplish this in polynomial time by discretizing the work curve, using a technique from [2, Section 2.6]. Specifically, we pre-round each machine speed down to the nearest power of $1 + \delta$ before running our algorithm. It is easy to verify that this does not affect monotonicity, nor does it affect our PTAS guarantee when δ is a suitably small constant. Moreover, the work curve of the new algorithm is discrete, and changes its value only at powers of $1 + \delta$.

To compute player i's payment in polynomial time, a difficulty remains: we need to show that evaluating the work curve at a polynomial number of speeds s'_i suffices for computing its integral. This is indeed the case, by the following observation: Let $p_{\min}$ and $p_{\max}$ denote the smallest and largest job sizes, respectively, and observe that the our algorithm never assigns any machine a non-zero amount of expected work smaller than $p_{\min}$; this follows from properties (P2) and (P3) of permissible partitions, assuming δ is sufficiently small. The approximation guarantee of $(1 + \epsilon)$ therefore ensures that no machine more than a $(1 + \epsilon)np_{\max}/p_{\min}$ factor slower than the fastest receives non-zero work. This in turn implies that, with pre-rounded speeds and for fixed reports s_{-i}, we can infer the expected work assigned to machine i for every alternative report from the results for a polynomial number of such reports s'_i (the powers of $(1 + \delta)$ in the appropriate range). We can obtain these in polynomial time by simply rerunning the third step of the generic algorithm (Algorithm 10.1) for each such report. The formula (A.1) for appropriate payments is then easy to compute exactly for each player i in polynomial time.

10.3 Additional Results

Variants of the algorithmic and analytical approach in Section 10.2 yield a number of additional results: a deterministic QPTAS for $Q||C_{\max}$ (Section 10.3.1); a randomized PTAS and deterministic QPTAS for minimizing the p-norm of loads on related machines (Section 10.3.2); and a randomized PTAS for maximizing the minimum load on related machines (Section 10.3.3). Throughout this section, we omit details that are essentially redundant with Section 10.2 and highlight only the main new ideas needed to obtain the claimed results.

10.3.1 A Deterministic Monotone QPTAS for Minimizing Makespan

We can apply our generic algorithm (Algorithm 10.1) to obtain easily a monotone deterministic quasipolynomial-time approximation scheme (QPTAS) for minimizing makespan.

Theorem 10.3.1. *There is a deterministic monotone QPTAS for $Q||C_{\max}$.*

Proof. Fix a set of n jobs and parameters m, δ, and set $l = \lceil \log_{(1+\delta)}(m/\delta) \rceil$. Let $\mathcal{S}$ denote the non-decreasing speed vectors s with $s_m = 1$ and with each s_i either 0 or of the form $(1+\delta)^{-k_i}$ for an integer k_i between 0 and l. There is a quasipolynomial number $m^{O(l)}$ of such speed vectors. Compute a $(1+\delta)$-approximate schedule for each using a (non-monotone) PTAS for $Q||C_{\max}$ such as [50] or [39], and let $\mathcal{X}$ denote the induced set of (integral) job partitions. We can explicitly construct and optimize over $\mathcal{X}$ in quasipolynomial time. Order $\mathcal{X}$ lexicographically by sorted work vectors, as in Theorem 10.2.11.

By Lemmas 10.2.2 and 10.2.6, we can complete the proof by showing that $\mathcal{X}$ is $O(\delta)$-good. Consider an arbitrary speed vector s; renaming and scaling, we can assume that s is non-decreasing with $s_m = 1$. Call machine i *slow* if $s_i < \delta/m$. Obtain the speed vector $\widehat{s}$ by zeroing out the speeds of slow machines and rounding all other speeds down to the nearest integer power of $(1+\delta)^{-1}$. The optimal makespan for speeds $\widehat{s}$ is at most $1 + O(\delta)$ times that for s (in proof, take an optimal schedule for s

and reassign jobs on slow machines to machine m). By construction, $\mathcal{X}$ contains a partition inducing a $(1+\delta)$-approximate schedule for $\widehat{s}$; this schedule is $(1+O(\delta))$-approximate for s. □

The exponential dependence on $\log^2 m$ in Theorem 10.3.1 improves upon the exponential dependence on m in the deterministic monotone algorithm of Andelman et al. [1].

10.3.2 Minimizing a Norm of Machine Loads

We can also extend our results to other parallel related machine scheduling problems. We first consider the problem of minimizing the p-norm of the machine loads (for $p \in [1, \infty]$). The obvious modification of our generic algorithm (Algorithm 10.1), in which we replace the makespan objective in the third step by that of minimizing the p-norm, remains monotone. The proof of Lemma 10.2.2 requires some modifications, as follows.

Proof. (of Lemma 10.2.2, adapted to minimizing the p-norm.) Let $s = (s_i, s_{-i})$ and $\widehat{s} = (\widehat{s}_i, s_{-i})$ denote two speed vectors that differ only for machine i, with $s_i > \widehat{s}_i$, and let $P, \widehat{P} \in \mathcal{X}$ denote the corresponding optimal partitions. Let machine i be the kth slowest in s and the $\widehat{k}$th slowest in $\widehat{s}$, with $\widehat{k} \leq k$. Assume for contradiction that $|P_k| < |\widehat{P}_{\widehat{k}}|$, so $|P_\ell| \leq |P_k| < |\widehat{P}_{\widehat{k}}| \leq |\widehat{P}_\ell|$ for each $\ell \in \{\widehat{k}, \ldots, k\}$.

Let $s(\ell), \widehat{s}(\ell)$ denote the speeds of the ℓth slowest machines in s and $\widehat{s}$, respectively. Switching from speeds s to $\widehat{s}$ increases the pth power of the p-norm of the schedule induced by P by

$$\sum_{\ell=\widehat{k}}^{k} |P_\ell|^p \left(\widehat{s}(\ell)^{-p} - s(\ell)^{-p}\right)$$

and that of the schedule induced by $\widehat{P}$ by

$$\sum_{\ell=\widehat{k}}^{k} |\widehat{P}_\ell|^p \left(\widehat{s}(\ell)^{-p} - s(\ell)^{-p}\right).$$

Thus the p-norm of the latter schedule increases at least as much as the former, contradicting the assumption that $\widehat{P}$ is the $\prec$-minimum optimal schedule for $\widehat{s}$. □

The proofs of Theorems 10.3.1 and 10.2.11 then carry over with only cosmetic changes. For example, for the analog of Theorem 10.2.11, we define permissible partitions as in Section 10.2. Lemma 10.2.9 remains valid because its proof extracts a permissible partition from an arbitrary integral schedule while increasing the work assigned to each machine, and hence the p-norm, by a $1+O(\delta)$ factor. Lemma 10.2.10 requires only trivial modifications and truthful payments can be computed as in Section 10.2.5.

Theorem 10.3.2. *There is a deterministic monotone QPTAS for minimizing the p-norm on related machines.*

Theorem 10.3.3. *There is a randomized monotone PTAS for minimizing the p-norm on related machines.*

10.3.3 Maximizing the Minimum Machine Load

Finally, we consider the problem of maximizing the minimum load on related machines. Again, the natural variant of our generic algorithm, using the max-min objective in the third step, is monotone; the proof is very similar to that of Lemma 10.2.2. Theorem 10.3.1 does not obviously extend to max-min scheduling, as the reassignment procedure (from slow machines to the fastest one) used in the proof need not produce a near-optimal solution. We can, however, extend Theorem 10.2.11 to max-min scheduling via some non-trivial modifications.

The main difficulty in extending the proof of Theorem 10.2.11 to the max-min objective is that removing jobs from machines, as in the reservation procedure in the proof of Lemma 10.2.9, can destroy near-optimality. To circumvent this problem, we relax our definition of permissible partitions. First, we insist only on 2δ-integrality rather than δ-integrality. For magnitudes w to be legal for a partition P, we require that w_i is at least $1/\delta$ times the full size of every job that is fractionally assigned in P_i for every $i < m$; and that w_m is at least $1/2\delta$ times the full size of every job fractionally assigned in P_m (cf., property (P2)). Finally, we replace property (P5) by:

(P5') for every block endpoint P_i other than P_m, either the induced w_i-configuration of S_i is integral, or else S_i includes all w_i-small jobs, where $S_i = \cup_{k\,:\,w_k \leq w_i} P_k$.

Next, we show how to modify the proof of Lemma 10.2.9 to establish that this relaxed set of permissible partitions is $O(\delta)$-good for the max-min objective.

Proof. (of Lemma 10.2.9, adapted to max-min scheduling.) Fix a job set, machine speeds s, and an optimal schedule σ^*. Let $(S_1, \ldots, S_m)$ denote the corresponding bundles. We can assume that $s_1 \leq \cdots \leq s_m$ and, by an exchange argument, that $|S_1| \leq \cdots \leq |S_m|$. We extract from σ^* a permissible partition, in the current relaxed sense, with minimum load (with respect to s) at least $1 - O(\delta)$ times that of σ^*.

For each i, define w_i as the unique power of 2 with $w_i/2 < |S_i| \leq w_i$; these are legal for the job partition induced by the schedule. We begin by iterating through the magnitudes W occurring in w in increasing order. We fractionally re-assign W-small jobs from machines with magnitude larger than W to those with magnitude equal to W, until the total fractional amount of W-small jobs assigned to the latter machines is either a multiple of δW or is all W-small jobs. This procedure enforces a strengthened form of property (P5'), and it removes at most δw_i work from each machine i (at most δW in each iteration with $W < w_i$).

To establish (P4), we again iterate through the magnitudes W of w, in arbitrary order. As in the proof of Lemma 10.2.9, we can re-assign W-small-jobs between machines with magnitude W until only one such machine remains with a non-integral W-configuration. Re-assigning again if needed, we can assume that this machine is the most heavily worked one with magnitude W (and thus will be a block endpoint provided there is a W-block). These re-assignments do not affect any previously established properties. The job partition induced by the resulting schedule satisfies property (P4), except for machines i that belong to the $(w_i/2)$-block of the partition and are the most heavily worked machine with magnitude w_i. For each such machine i, we re-assign the minimal (fractional) amount of w_i-small jobs to the next block endpoint with magnitude exceeding w_i. This is always possible unless w_i is the largest magnitude; in this case, we move the same amount to the most heavily worked machine k. Our relaxed version of property (P2) allows this, even in the event that w_k

is only $w_i/2$. (Note that if we reindex machines according to their new bundle sizes, machine k corresponds to machine m in property (P2).) No machine i loses more than δw_i work in this second round of re-assignments.

Finally, we restore 2δ-integrality of the job partition by re-assigning small jobs as at the end of Section 10.2.3. □

Modifying the representation and algorithm in Section 10.2.4 to accommodate this wider set of permissible partitions is relatively straightforward. Parameters $W_1, W_2, A_1, B_1, A_2, B_2$ are now *valid* if each of B_1, A_2, B_2 is integral, A_1 is either integral or represents a superset of all W_1-small jobs, and all subsets of $\{A_1, A_2, B_1, B_2\}$ sum to realizable configurations. Crucially, there are still only polynomially many valid sets of parameters. As in Section 10.2.4, given machine speeds, the optimal permissible partition can be found by an s-t path computation.

Theorem 10.3.4. *There is a randomized monotone PTAS for maximizing the minimum load on related machines.*

Truth-telling payments exist for this monotone PTAS, as usual. However, unlike all other problems studied in this chapter, the form of the max-min objective requires any algorithm with a non-zero approximation ratio to assign non-zero work to every machine, no matter how slow. Furthermore, when the approximation ratio does not depend on player reports, the induced "work curve" of a player (Figure A.1) is lower-bounded by an inverse function of the player's report. Consequently the integral of the work curve is divergent. As described in Appendix A.3, this implies that no truth-telling payment rule guarantees individual rationality nor the proper sign for the payments. That being said, the generalized form of the payment rule (Equation (A.2)) with the pivot terms set to 0 suffices for truthfulness in expectation alone, and can be computed in polynomial time using the same ideas as those used in Section 10.2.5.

Part V

Conclusion

Chapter 11

Conclusions and Open Questions

The work in this thesis investigates efficient allocation of resources in large systems that involve strategic actors. Our discoveries are motivated by the tension between the economic goal, that of designing incentives so that strategic behavior leads to a desired global outcome, and the computational goal of doing so efficiently. We discover that careful use of randomization can reconcile economic and computational goals in settings where doing so was previously thought to be untenable due to the failure of existing — often deterministic — techniques. In addition to improving the state of the art for several fundamental resource allocation problems, we believe that a primary contribution of our work is a toolbox for the design of efficient approximation mechanisms for resource allocation.

We conclude with some open questions motivated by the results of this thesis. We begin with a concrete question motivated by the results of Part II.

> **Open Question:** *Are there constant-factor approximation mechanisms for more general variants of combinatorial auctions and combinatorial public projects than those considered in Chapters 5 and 6?*

Our results for both problems apply to players with Matroid Rank Sum valuations. Are more general results possible? First, what is possible for other valuation classes for which these problems admit constant-factor approximation algorithms, such as submodular functions? Work since our results in [26, 38] has ruled out constant-factor approximation mechanisms that are truthful in expectation for combinatorial

auctions and public projects with submodular valuations in the *value oracle model*; understanding the extent of these limitations under more relaxed assumptions on the presentation of the valuations is, however, still open. In particular, the results of [26, 38] proceed by proving that a constant-factor approximation mechanism for combinatorial auctions or public projects requires the solution of an intractable single-player utility maximization problem, motivating the following question: what if we assume that "players know what they want"? The demand oracle model, described in Section 5.2.2, makes such an assumption. We leave open the question of whether our results can be extended to submodular valuations, or any of the other valuation classes naturally considered in a combinatorial auctions context, when players can answer more detailed queries regarding their individual valuations.

Second, we mention a different and natural extension of combinatorial public projects. Consider a feasible set where the set of projects built is not subject to a cardinality constraint, but rather a different packing-type constraint. For instance, in a setting where an exogenous set of resources must be matched to the chosen set of projects, and a bipartite graph encodes the possible resource-project matchings, the set of projects chosen by the public planner must satisfy a *matroid constraint* — in this example corresponding to a transversal matroid.[1] We propose studying combinatorial public projects with matroid rank sum valuations and a matroid constraint on the feasible set of projects chosen; it is conceivable that a convex rounding scheme exists for this problem that yields a constant-factor approximation mechanism.

Third, the results of Part II motivate a more general question regarding the strength and limitations of convex rounding.

> **Open Question:** *Are there large classes of linear programs, or mathematical relaxations more generally, that always admit a convex rounding algorithm that matches (or comes close to) the approximation guarantee of the best (possibly non-convex) rounding algorithm?*

[1] In a practical example, a project is a computational task to be scheduled on a grid, and a resource is a machine capable of performing a single task, and the existence of an edge in the task-machine bipartite graph indicates that the machine has the necessary software and/or hardware to perform the task.

A positive answer to this question would be technically and conceptually remarkable. It would, for a large class of problems, presumably provide a generic procedure that converts an arbitrary approximation algorithm in the relax-solve-round framework to an incentive-compatible mechanism with the same approximation guarantee. While not a black-box result in the sense of Chapter 8, this would arguably be the "next best thing".

Fourth, motivated by Chapter 8, we next turn the possibility of more general black box reductions.

> **Open Question:** *Are more general black-box reductions from incentive-compatible mechanism design to approximation algorithm design possible?*

Such a reduction would convert an arbitrary approximation algorithm to an incentive-compatible mechanism in polynomial time, without much degrading its approximation guarantee. Whereas recent impossibility results like those in [26, 38] rule out a "holy grail" reduction that is arbitrarily general, such results may still exist for large and natural classes of problems.

A natural class of problems for which black box reductions are conceivable is the class of *linear optimization problems* with a welfare objective, treated in Part III. We recall that the black-box reduction of Chapter 8 applies to many of these problems when they admit an FPTAS. It would be remarkable to extend this result to approximation algorithms with a weaker approximation guarantee, in the process showing that incentive compatibility is "without loss" for these problems. Another candidate is the class of *single-parameter problems* studied in Chapter 10. Several indications point to viability of general positive results for this class: the space of incentive-compatible mechanisms is larger and more forgiving for single-parameter problems than it is for their multi-parameter counterparts; optimal mechanisms for most examples of these problems in the literature have been designed over the past decade; and black-box results for these problems in more forgiving Bayesian settings have been recently discovered [48].

We close by noting that all the questions we pose are special cases of a broader, arguably somewhat vague, meta question:

> **Open Question:** *How powerful are computationally-efficient incentive-compatible mechanisms for resource allocation?*

Many interpretations of this question are possible, and we mention two here. One interpretation may seek to characterize the space of resource allocation problems for which incentive compatibility is "without loss" — namely, a computationally-efficient and incentive-compatible mechanism matches, or comes close to, the approximation guarantee of the best computationally-efficient algorithm for the problem. Specifically, what structural properties of a resource allocation problem make it amenable to the combination of incentive compatibility and computational efficiency, and what others make it resist such a solution? Another interpretation may seek to identify the best incentive-compatible and computationally-efficient mechanism for a resource allocation problem, stated abstractly. Specifically, what is the best incentive-compatible and computationally-efficient mechanism for a resource allocation problem, how well does it approximate its objective, and how does its approximation guarantee depend on the parameters of the problem?

Even a partial answer to any variant of this broad question appears to require a deep understanding of the space of incentive-compatible mechanisms, polynomial-time algorithms, and their intersection. We have only scratched the surface.

Part VI

Appendices

Appendix A

Additional Preliminaries

A.1 Matroid Theory

In this section, we review some basics of matroid theory. For a more comprehensive reference, we refer the reader to [73]. A *matroid* M is a pair $(\mathcal{X}, \mathcal{I})$, where $\mathcal{X}$ is a finite *ground set*, and $\mathcal{I}$ is a non-empty family of subsets of $\mathcal{X}$ satisfying the following two properties. (1) *Downward closure:* If S belongs to $\mathcal{I}$, then so do all subsets of S. (2) *The exchange property:* Whenever $T, S \in \mathcal{I}$ with $|T| < |S|$, there is some $x \in S \setminus T$ such that $T \cup \{x\} \in \mathcal{I}$. Elements of $\mathcal{I}$ are often referred to as the *independent sets* of the matroid. Subsets of $\mathcal{X}$ that are not in $\mathcal{I}$ are often called *dependent*.

We associate with matroid M a set function $rank_M : 2^{\mathcal{X}} \to \mathbb{N}$, known as the *rank function of* M, defined as follows: $rank_M(A) = \max_{S \in \mathcal{I}} |S \cap A|$. Equivalently, the rank of set A in matroid M is the maximum size of an independent set contained in A. A set function f on a ground set $\mathcal{X}$ is a *matroid rank function* if there exists a matroid M on the same ground set such that $f = rank_M$. Matroid rank functions are non-decreasing ($f(S) \leq f(T)$ when $S \subseteq T$), normalized ($f(\emptyset) = 0$), and submodular ($f(S) + f(T) \geq f(S \cap T) + f(S \cup T)$ for all S and T).

For a matroid $M = (\mathcal{X}, \mathcal{I})$ and $S \subseteq \mathcal{X}$, we define the *contraction* of M by S, denoted by M/S. M/S is a pair $(\mathcal{X} \setminus S, \mathcal{I}')$, where $\mathcal{I}'$ is the following family of subsets of $\mathcal{X} \setminus S$: A set $T \subseteq \mathcal{X} \setminus S$ is in $\mathcal{I}'$ if and only if $rank_M(S \cup T) - rank_M(S) = |T|$. For each matroid $M = (\mathcal{X}, \mathcal{I})$ and $S \subseteq \mathcal{X}$, the contraction M/S is also a matroid.

A.2 Convex Optimization

In this section, we distill some basics of convex optimization. For more details, see [10].

Definition A.2.1. *A Euclidean maximization problem is a maximization problem* Π *(as defined in Section 2.3) where for each instance* $(\mathcal{P}, c) \in \Pi$ *the feasible set* $\mathcal{P}$ *is a subset of some Euclidean space. We say* Π *is a convex maximization problem if for every* $(\mathcal{P}, c) \in \Pi$*,* $\mathcal{P}$ *is a compact convex set, and* $c : \mathcal{P} \to \mathbb{R}$ *is concave.*

Definition A.2.2. *We say a non-negative Euclidean maximization problem* Π *is* R-solvable *in polynomial time if there is an algorithm that takes as input the representation of an instance* $\mathcal{I} = (\mathcal{P}, c) \in \Pi$ *— where we use* $|\mathcal{I}|$ *to denote the number of bits in the representation — and an approximation parameter* ϵ*, and in time* $\mathrm{poly}(|\mathcal{I}|, \log(1/\epsilon))$ *outputs* $x \in \mathcal{P}$ *such that* $c(x) \geq (1-\epsilon)\max_{y\in\mathcal{P}} c(y)$*.*

Fact A.2.3. *Consider a non-negative convex maximization problem* Π*. If the following four conditions are satisfied, then* Π *is* R*-solvable in polynomial time using the ellipsoid method. We let* $\mathcal{I} = (\mathcal{P}, c)$ *denote an instance of* Π*, and let* m *denote the dimension of the ambient Euclidean space.*

1. *Polynomial Dimension:* m *is polynomial in* $|\mathcal{I}|$*.*
2. *Starting ellipsoid: There is an algorithm that computes, in time* $\mathrm{poly}(|\mathcal{I}|)$*, a point* $c \in \mathbb{R}^m$*, a matrix* $A \in \mathbb{R}^{m\times m}$*, and a number* $\mathcal{V} \in \mathbb{R}$ *such that the following hold. We use* $E(c, A)$ *to denote the ellipsoid given by center* c *and linear transformation* A*.*
   (a) $E(c, A) \supseteq \mathcal{P}$
   (b) $\mathcal{V} \leq volume(\mathcal{P})$
   (c) $\frac{volume(E(c,A))}{\mathcal{V}} \leq 2^{\mathrm{poly}(|\mathcal{I}|)}$
3. *Separation oracle for* $\mathcal{P}$*: There is an algorithm that takes takes input* $\mathcal{I}$ *and* $x \in \mathbb{R}^m$*, and in time* $\mathrm{poly}(|\mathcal{I}|, |x|)$ *where* $|x|$ *denotes the size of the representation of* x*, outputs "yes" if* $x \in \mathcal{P}$*, otherwise outputs* $h \in \mathbb{R}^m$ *such that* $h^T x < h^T y$ *for every* $y \in \mathcal{P}$*.*

4. *First order oracle for c: There is an algorithm that takes input* $\mathcal{I}$ *and* $x \in \mathbb{R}^m$*, and in time* $\text{poly}(|\mathcal{I}|, |x|)$ *outputs* $c(x) \in \mathbb{R}$ *and* $\bigtriangledown c(x) \in \mathbb{R}^m$*.*

A.3 Single-Parameter Mechanism Design

In this section, we review some basic facts from prior work on single-parameter mechanism design, and provide intuition regarding their veracity. For consistency with Chapter 10, we define single-parameter mechanism design problems where each player has a private *cost function* over the outcomes, though note that this presentation is merely a sign-flip away from the *valuation function* convention adopted in most of the rest of this thesis.

Recall from Chapter 10 that a mechanism design problem with m players is *single-parameter* if all outcomes $\omega \in \Omega$ are real m-vectors, and each player i's private cost function has the form $t_i(\omega) = c_i\omega_i$ for a private real number c_i. For an outcome ω and player i, we call ω_i the *work* assigned to player i by ω.

Let $\mathcal{A}$ be an allocation rule mapping private reports $c \in \mathbb{R}^m$ to outcomes $\omega \in \Omega$, and let $x_i(c)$ denote the expected work assigned to player i by $\mathcal{A}(c)$, where the expectation is taken over the internal random coins of $\mathcal{A}$. We say $\mathcal{A}$ is *monotone* if for each player i and fixed reports c_{-i} of the other players, $x_i(c_i, c_{-i})$ is a non-increasing function of c_i. The following characterization is implicit in the work of Mirrlees [66] and Spence [82]. Later, Myerson [68] makes this characterization explicit in a Bayesian context, and subsequently Archer and Tardos [3] do the same for dominant-strategy implementation.

Theorem A.3.1 ([66, 82, 68, 3])**.** *An allocation rule for a single parameter problem is implementable by a truthful-in-expectation mechanism if and only if it is monotone.*

For completeness, we provide intuition regarding the veracity of the positive direction of this characterization, and present the form of the truth-telling payment rule. For a more detailed and formal treatment, we recommend [2].

Fix allocation rule $\mathcal{A} : \mathbb{R}^m \to \Omega$, and let $x : \mathbb{R}^m \to \mathbb{R}^m$ be its expected work

function as described above. Consider the rule that awards player i the following payment on reports $c \in \mathbb{R}^m$.

$$p_i(c) = c_i x_i(c) + \int_{b=c_i}^{\infty} x_i(b, c_{-i}) db \tag{A.1}$$

When these payments are defined and $\mathcal{A}$ is monotone, $(\mathcal{A}, p)$ is truthful in expectation.

Proposition A.3.2 ([66, 82, 68, 3])**.** *Assume that the integral in Equation* (A.1) *converges for every c. If $\mathcal{A}$ is monotone, then mechanism $(\mathcal{A}, p)$ is truthful in expectation, individually rational in expectation, and its payments to the players are always non-negative.*

The assumption that the integral in Equation (A.1) converges is justified in most applications, as we would expect a "typical" allocation rule to eventually decrease player i's work to 0 as his cost per unit of work is increased. For such a "typical" allocation rule, Equation (A.1) admits a natural interpretation. We describe this interpretation and use it to informally argue Proposition A.3.2 next.

Fix a player i and reports c_{-i} of the other players. Let $b_{\max}$ be the report at which player i is assigned no work, i.e. $x_i(b_{max}, c_{-i}) = 0$. When player i reports c_i, his payment $p_i(c)$ is the outcome of the following thought experiment, illustrated in Figure A.1: Consider a hypothetical report b for player i, and slowly decrease b from its maximum "relevant" value $b_{\max}$ to the actual reported value c_i. As b is decreased, compensate i incrementally at rate b for each additional slice of work he receives. The rate of compensation for a particular slice is independent of the player's report, and equals the maximum report at which the player would receive said slice. In the resulting mechanism, the player in effect faces a "take it or leave it" offer for each slice of work, and "takes" all those slices of work for which his reported cost is no greater than his compensation. It is in the best interest of the player to "take" all the slices for which his true cost is no greater than his compensation, and therefore this mechanism incentivizes truth-telling. Individual rationality and the direction of the payments also follow. This interpretation reveals that this mechanism is a continuous generalization of the truthful Vickrey auction (Section 1.2.1), in which a player faces

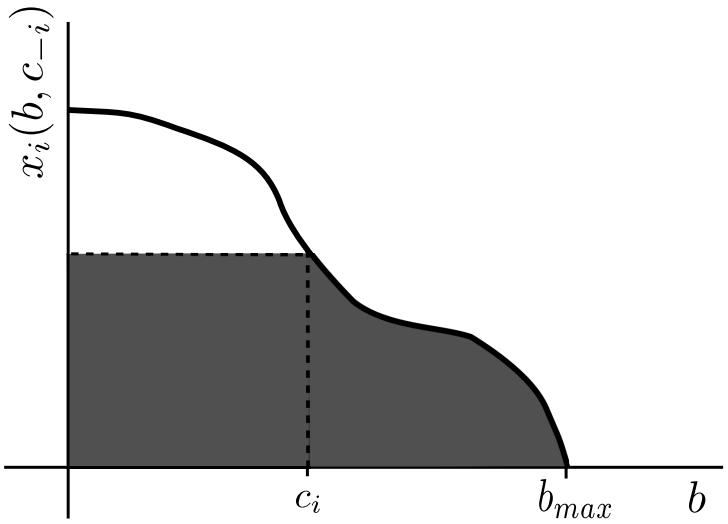


Figure A.1: Truth-telling payments for a monotone allocation rule. The curve represents player i's expected work as a function of his report b, holding the reports c_{-i} of other players fixed. The payment to player i when he reports c_i is equal to area of the shaded region.

a "take it or leave it" price equal to the minimum bid needed to win. This concludes our discussion of Proposition A.3.2.

Finally, we note that the form of the payments in Equation (A.1) is a special case of a more flexible payment rule. In particular, the following payment rule, parametrized by a bid-independent "pivot term" $h_i(c_{-i})$ for each player i, is a generalization:

$$p_i(c) = h_i(c_{-i}) + c_i x_i(c) - \int_{b=0}^{c_i} x_i(b, c_{-i}) db \tag{A.2}$$

Note that setting $h_i(c_{-i}) = \int_{b=0}^{\infty} x_i(b, c_{-i})$ recovers Equation (A.1). The generalized form (A.2) of the payment rule is useful when Equation (A.1) is not guaranteed to converge, as is unavoidable for one of the scheduling problems considered in Chapter 10. For such problems, we can set $h_i(c_{-i}) = 0$, sacrificing individually rationality and allowing payments from the players to the mechanism, but preserving truthfulness in expectation.

Appendix B

Omitted Proofs

B.1 Solving The Convex Program of Chapter 5

In this section, we overcome technical difficulties related to the solvability of the convex program of Chapter 5. We show in Section B.1.1 that, in the lottery-value oracle model, the four conditions for "solvability" of convex programs, as stated in Fact A.2.3, are easily satisfied for convex program (5.3). However, an additional challenge remains: "solving" a convex program – as in Definition A.2.2 – returns an approximately optimal solution. Indeed the optimal solution of a convex program may be irrational in general, so this is unavoidable.

We show how to overcome this difficulty if we settle for polynomial runtime in expectation. While the optimal solution x^* of (5.3) cannot be computed explicitly, the random variable $r_{\text{poiss}}(x^*)$ can be sampled in expected polynomial time. The key idea is the following: *sampling the random variable* $r_{\text{poiss}}(x^*)$ *rarely requires precise knowledge of* x^*. Depending on the coin flips of r_{poiss}, we decide how accurately we need to solve convex program (5.3) in order compute $r_{\text{poiss}}(x^*)$. Roughly speaking, we show that the probability of requiring a $(1-\epsilon)$-approximation falls exponentially in $\frac{1}{\epsilon}$. As a result, we can sample $r_{\text{poiss}}(x^*)$ in expected polynomial time. We implement this plan in Section B.1.2 under the simplifying assumption that convex program (5.3) is *well-conditioned* – i.e. is "sufficiently concave" everywhere. In Section B.1.3, we show how to remove that assumption by slightly modifying our algorithm.

B.1.1 Approximating the Convex Program

Claim B.1.1. *There is an algorithm for Combinatorial Auctions with MRS valuations in the lottery-value oracle model that takes as input an instance of the problem and an approximation parameter $\epsilon > 0$, runs in $\mathrm{poly}(n, m, \log(1/\epsilon))$ time, and returns a $(1-\epsilon)$-approximate solution to convex program (5.3).*

It suffices to show that the four conditions of Fact A.2.3 are satisfied in our setting. The first three are immediate from elementary combinatorial optimization (see for example [56]). It remains to show that the first-order oracle, as defined in Fact A.2.3, can be implemented in polynomial time in the lottery-value oracle model. The objective $f(x)$ of convex program (5.3) can, by definition, be written as

$$f(x) = \sum_i G_{v_i}(x_i),$$

where v_i is the valuation function of player i, x_i is the vector $(x_{i1}, \ldots, x_{im})$, and and G_{v_i} is as defined in (5.6). By definition, $G_{v_i}(x_i)$ is the outcome of querying the lottery-value oracle of player i with $(1-e^{-x_{i1}}, \ldots, 1-e^{-x_{im}})$. Therefore, we can evaluate $f(x)$ using n lottery-value query, one for each player. It remains to show that we can also evaluate the (multi-variate) derivative $\bigtriangledown f(x)$ of $f(x)$. Using definition (5.6), we take the partial derivative corresponding to x_{ij}. By rearranging the sum appropriately, we get that

$$\frac{\partial f}{\partial x_{ij}}(x) = e^{-x_{ij}}\Big(F_{v_i}\big((1-e^{-x_{i1}}, \ldots, 1-e^{-x_{im}}) \vee 1_j\big) - F_{v_i}\big((1-e^{-x_{i1}}, \ldots, 1-e^{-x_{im}}) \wedge 0_j\big)\Big),$$

where F_{v_i} is as defined in Equation (5.1). Here, $\vee$ and $\wedge$ denote entry-wise minimum and maximum respectively, 1_j denotes the vector with all entries equal to 0 except for a 1 at position j, and 0_j denotes the vector with all entries equal to 1 except for a 0 at position j. It is clear that this entry of the gradient of f can be evaluated using two lottery-value queries. Therefore, $\bigtriangledown f(x)$ can be evaluated using $2n$ lottery-value queries, 2 for each player. This completes the proof of Claim B.1.1.

B.1.2 The Well-Conditioned Case

In this section, we make the following simplifying assumption: The objective function $f(x)$ of convex program (5.3), when restricted to any line in the feasible set $\mathcal{P}$, has a second derivative of magnitude at least $\lambda = \frac{\sum_{i=1}^{n} v_i([m])}{2^{\mathrm{poly}(n,m)}}$ everywhere, where the polynomial in the denominator may be arbitrary. This is equivalent to requiring that every eigenvalue of the Hessian matrix of $f(x)$ has magnitude at least λ when evaluated at any point in $\mathcal{P}$. Under this assumption, we prove Lemma B.1.2.

Lemma B.1.2. *Assume the magnitude of the second derivative of $f(x)$ is at least $\lambda = \frac{\sum_{i=1}^{n} v_i([m])}{2^{\mathrm{poly}(n,m)}}$ everywhere. Algorithm 4.1, instantiated for combinatorial auctions with $r = r_{\mathrm{poiss}}$, can be simulated in time polynomial in n and m in expectation.*

Let x^* be the optimal solution to convex program (5.3). Algorithm 4.1 allocates items according to the distribution $r_{\mathrm{poiss}}(x^*)$. The Poisson rounding scheme, as described in Algorithm 5.1, requires making m independent decisions, one for each item j. Therefore, we fix item j and show how to simulate this decision. It suffices to do the following in expected polynomial time: flip uniform coin $p_j \in [0,1]$, and find the minimum index $a(j)$ (if any) such that $\sum_{i \le a(j)} (1 - e^{-x^* ij}) \ge p_j$. For most realizations of p_j, this can be decided using only coarse estimates $\widetilde{x}_{ij}$ to x^*_{ij}. Assume we have an *estimation oracle* for x^* that, on input δ, returns a *δ-estimate* $\widetilde{x}$ of x^*: Specifically, $\widetilde{x}_{ij} - x^*_{ij} \le \delta$ for each i. When p_j falls outside the "uncertainty zones" of $\widetilde{x}$, such as when $|p_j - \sum_{i' \le i}(1 - e^{-\widetilde{x}_{i'j}})| > \delta n$ for each $i \in [n]$, it is easy to see that we can correctly determine $a(j)$ by using $\widetilde{x}$ in lieu of x. The total measure of the uncertainty zones of $\widetilde{x}$ is at most $2n^2\delta$, therefore p_j lands outside the uncertainty zones with probability at least $1 - 2n^2\delta$. The following claim shows that if the estimation oracle for x^* can be implemented in time polynomial in $\log(1/\delta)$, then we can simulate the Poisson rounding procedure in expected polynomial time.

Claim B.1.3. *Let x^* be the optimal solution of convex program* (5.3)*. Assume access to a subroutine $B(\delta)$ that returns a δ-estimate of x^* in time* $\mathrm{poly}(n, m, \log(1/\delta))$*. Algorithm* (4.1) *with $r = r_{\mathrm{poiss}}$ can be simulated in expected* $\mathrm{poly}(n, m)$ *time.*

Proof. It suffices to show that we can simulate the allocation of an item j by Algorithm (5.1) on input x^*. The simulation proceeds as follows: Draw $p_j \in [0,1]$

uniformly at random. Start with $\delta = \delta_0 = \frac{1}{2n^2}$. Let $\widetilde{x} = B(\delta)$. While $|p_j - \sum_{i' \leq i}(1 - e^{-\widetilde{x}_{i'j}})| \leq \delta n$ for some $i \in [n]$ (i.e. p_j may fall inside an "uncertainty zone") do the following: let $\delta = \delta/2$, $\widetilde{x} = B(\delta)$ and repeat. After the loop terminates, we have a sufficiently accurate estimate of x^* to calculate $a(j)$ as in Algorithm (5.1).

It is easy to see that the above procedure is a faithful simulation of Algorithm (5.1) on x^*. It remains to bound its expected running time. Let $\delta_k = \frac{1}{2^{k+1}n^2}$ denote the value of δ at the kth iteration. By assumption, the kth iteration takes time polynomial in n, m and $\log(1/\delta_k)$, which by definition of δ_k is polynomial in n, m, and k. The probability this procedure does not terminate after k iterations is at most $2n^2\delta_k = 1/2^k$. Taken together, these two facts and a simple geometric summation imply that the expected runtime is polynomial in n and m. □

It remains to show that the estimation oracle $B(\delta)$ can be implemented in time polynomial in n, m, and $\log(1/\delta)$. At first blush, one may expect that the ellipsoid method can be used in the usual manner here. However, there is one complication: we require an estimate $\widetilde{x}$ that is close to x^* *in solution space* rather than in terms of objective value. Using our assumption on the curvature of $f(x)$, we will reduce finding a δ-estimate of x^* to finding an $1-\epsilon(\delta)$ approximate solution to convex program (5.3). The dependence of ϵ on δ will be such that $\epsilon \geq \text{poly}(\delta)/2^{\text{poly}(n,m)}$, thereby we can invoke Claim B.1.1 to deduce that $B(\delta)$ can be implemented in $\text{poly}(n, m, \log(1/\delta))$ time.

Let $\epsilon = \epsilon(\delta) = \frac{\delta^2\lambda}{2\sum_i v_i([m])}$. Plugging in the definition of λ, we deduce that $\epsilon \geq \delta^2/2^{\text{poly}(n,m)}$, which is the desired dependence. It remains to show that if $\widetilde{x}$ is $(1-\epsilon)$-approximate solution to (5.3), then $\widetilde{x}$ is also a δ-estimate of x^*.

Using the fact that $f(x)$ is concave, and moreover its second derivative has magnitude at least λ, it a simple exercise to bound distance of any point x from the optimal point x^* in terms of its sub-optimality $f(x^*) - f(x)$, as follows:

$$f(x^*) - f(x) \geq \frac{\lambda}{2}||x - x^*||^2. \tag{B.1}$$

Assume $\widetilde{x}$ is a $(1-\epsilon)$-approximate solution to (5.3). Equation (B.1) implies that

$$||\widetilde{x} - x^*||^2 \leq \frac{2}{\lambda}\epsilon f(x^*) = \frac{\delta^2}{\sum_i v_i([m])} f(x^*) \leq \delta^2,$$

where the last inequality follows from the fact that $\sum_i v_i([m]))$ is an upper-bound on the optimal value $f(x^*)$. Therefore, $||x - x^*|| \leq \delta$, as needed. This completes the proof of Lemma B.1.2.

B.1.3 Guaranteeing Good Conditioning

In this section, we propose a modification r^+_{poiss} of the Poisson rounding scheme r_{poiss}. We will argue that r^+_{poiss} satisfies all the properties of r_{poiss} established so far, with one exception: the approximation guarantee of Lemma 5.4.2 is reduced to $1-1/e-2^{-2mn}$. Then we will show that r^+_{poiss} satisfies the curvature assumption of Lemma B.1.2, demonstrating that said assumption may be removed. Therefore Algorithm 4.1, instantiated with $r = r^+_{\text{poiss}}$ for combinatorial auctions with MRS valuations in the lottery-value oracle model, is $(1-1/e-2^{-2mn})$ approximate and can be implemented in expected $\text{poly}(n,m)$ time. Finally, we show in Remark B.1.4 how to recover the 2^{-2mn} term to get a clean $1-1/e$ approximation ratio, as claimed in Theorem 5.3.1.

Let $\mu = 2^{-2mn}$. We define r^+_{poiss} in Algorithm B.1. Intuitively, r^+_{poiss} at first makes a tentative allocation using r_{poiss}. Then, it cancels said allocation with small probability μ. Finally, with probability β it chooses a random "lucky winner" i^* and gives him all the items. β is defined as the fraction of items allocated in the original tentative allocation. The motivation behind this seemingly bizarre definition of r^+_{poiss} is purely technical: as we will see, it can be thought of as adding "concave noise" to r_{poiss}.

We can write the expected welfare as follows. We use linearity of expectations and the fact that β is independent of the choice of i^* to simplify the expression.

$$\begin{aligned}
\mathbf{E}[w(r^+_{\text{poiss}}(x))] &= \mathbf{E}[(1-\mu)w(r_{\text{poiss}}(x)) + \mu\beta v_{i*}([m])] \\
&= (1-\mu)\,\mathbf{E}[w(r_{\text{poiss}}(x))] + \mu\,\mathbf{E}[\beta]E[v_{i*}([m])] \\
&= (1-\mu)\,\mathbf{E}[w(r_{\text{poiss}}(x))] + \mu\,\mathbf{E}[\beta]\frac{\sum_i v_i([m])}{n}
\end{aligned}$$

Input: Fractional allocation x with $\sum_i x_{ij} \leq 1$ for all j, and $0 \leq x_{ij} \leq 1$ for all i, j.
Output: Feasible allocation $(S_1, \ldots, S_n)$.

Let $(S_1, \ldots, S_n) \sim r_{\text{poiss}}(x)$.
Let $\beta = \frac{\sum_i |S_i|}{m}$.
Draw $q_1 \in [0, 1]$ uniformly at random.
if $q_1 \in [0, \mu]$ **then**
    Let $(S_1, \ldots, S_n) = (\emptyset, \emptyset, \ldots, \emptyset)$.
    Draw $q_2 \in [0, 1]$ uniformly at random.
    if $q_2 \in [0, \beta]$ **then**
        Choose a player i^* uniformly at random.
        Let $S_{i^*} = [m]$, and $S_i = \emptyset$ for all $i \neq i^*$.
    end if
end if

Algorithm B.1: The modified Poisson rounding scheme r^+_{poiss}.

Observe that r_{poiss} allocates an item j with probability $\sum_i (1 - e^{-x_{ij}})$. Therefore, the expectation of β is $\frac{\sum_{ij}(1-e^{-x_{ij}})}{m}$. This gives:

$$\mathbf{E}[w(r^+_{\text{poiss}}(x))] = (1-\mu)\,\mathbf{E}[w(r_{\text{poiss}}(x))] + \frac{\mu}{mn}\sum_i v_i([m]) \sum_{i,j}(1 - e^{-x_{ij}}). \tag{B.2}$$

It is clear that the expected welfare when using $r = r^+_{\text{poiss}}$ is within $1-\mu = 1-2^{-2mn}$ of the expected welfare when using $r = r_{\text{poiss}}$ in the instantiation of Algorithm 4.1. Using Lemma 5.4.2, we conclude that r^+_{poiss} is a $(1-1/e-2^{-2mn})$-approximate rounding scheme. Moreover, using Lemma 5.4.1, as well as the fact that $(1 - e^{-x_{ij}})$ is a concave function, we conclude that r^+_{poiss} is a convex rounding scheme. Therefore, this establishes the analogues of Lemmas 5.4.2 and5.4.1 for r^+_{poiss}. It is elementary to verify that our proof of Lemma B.1.2 can be adapted to r^+_{poiss} as well.

It remains to show that r^+_{poiss} is "sufficiently concave". This would establish that the conditioning assumption of Section B.1.2 is unnecessary for r^+_{poiss}. We will show that expression (B.2) is a concave function with curvature of magnitude at least

$\lambda = \frac{\sum_{i=1}^{n} v_i([m])}{emn2^{2mn}}$ everywhere. Since the curvature of concave functions is always non-positive, and moreover the curvature of the sum of two functions is the sum of their curvatures, it suffices to show that the second term of the sum (B.2) has curvature of magnitude at least λ. We note that the curvature of $\sum_{ij}(1 - e^{-x_{ij}})$ is at least e^{-1} over $x \in [0,1]^{n\times m}$. Therefore, the curvature of the second term of (B.2) is at least

$$\frac{\mu}{mn}\left(\sum_i v_i([m])\right)e^{-1} = \lambda$$

as needed.

Remark B.1.4. *In this section, we sacrificed 2^{-2mn} in the approximation ratio in order to guarantee expected polynomial runtime of our algorithm even when convex program* (5.3) *is not well-conditioned. This loss can be recovered to get a clean $1-1/e$ approximation as follows. Given our $(1 - 1/e - 2^{-2mn})$-approximate MIDR algorithm $\mathcal{A}$, construct the following algorithm $\mathcal{A}'$: Given an instance of combinatorial auctions, $\mathcal{A}'$ runs $\mathcal{A}$ on the instance with probability $1 - e2^{-2mn}$, and with the remaining probability solves the instance optimally in exponential time $O(2^{2mn})$. By Lemma 3.4.2, a random composition of MIDR mechanisms is MIDR, therefore $\mathcal{A}'$ is MIDR. The expected runtime of A' is bounded by the expected runtime of $\mathcal{A}$ plus $e2^{-2mn} \cdot O(2^{2mn}) = O(1)$. Finally, the expected approximation of A' is the weighted average of the approximation ratio of $\mathcal{A}$ and the optimal approximation ratio 1, and is at least $(1 - e2^{-2mn})(1 - 1/e - 2^{-2mn}) + e2^{-2mn} \geq 1 - 1/e$.*

B.2 Solving The Convex Program of Chapter 6

In this section, we show how to solve the convex program of Chapter 6. We follow the general outline of Appendix B.1, modifying the proofs throughout to handle the additional technical difficulties specific to CPP. We show in Section B.2.1 that, in the bounded-lottery-value oracle model, the four conditions for "solvability" of convex programs (Fact A.2.3) are easily satisfied for convex program (6.2). However, as in Appendix B.1, the additional challenge of solving our convex program *exactly* remains.

As in Appendix B.1, we overcome this challenge by allowing our algorithm to run in expected polynomial time. The key idea is the same: While the optimal solution x^* of (6.2) cannot be computed explicitly, the random variable $r_k(x^*)$ can be sampled in expected polynomial time. Depending on the coin flips of r_k, we decide how accurately we need to solve convex program (6.2) in order compute $r_k(x^*)$. Roughly speaking, we show that the probability of requiring a $(1-\epsilon)$-approximation falls exponentially in $\frac{1}{\epsilon}$. As a result, we can sample $r_k(x^*)$ in expected polynomial time. We implement this plan in Section B.2.2 under the simplifying assumption that convex program (6.2) is *well-conditioned* — i.e. is "sufficiently concave" everywhere. In Section B.2.3, we show how to remove that assumption by slightly modifying our algorithm.

B.2.1 Approximating the Convex Program

Claim B.2.1. *There is an algorithm for Combinatorial Public Projects with MRS valuations in the bounded-lottery-value oracle model that takes as input an instance of the problem and an approximation parameter $\epsilon > 0$, runs in $\text{poly}(n, m, \log(1/\epsilon))$ time, and returns a $(1-\epsilon)$-approximate solution to convex program (6.2).*

It suffices to show that the four conditions of Fact A.2.3 are satisfied in our setting. The first three are immediate from elementary combinatorial optimization (see for example [56]). It remains to show that the first-order oracle, as defined in Fact A.2.3, can be implemented in polynomial time in the bounded-lottery-value oracle model. We let $f(x)$ denote the objective function of convex program (6.2). This objective can, by definition, be written as follows.

$$f(x) = \mathop{\mathbf{E}}_{S \sim r_k(x)} \left[\sum_i v_i(S) \right] = \sum_i G_k^{v_i}(x)$$

where v_i is the valuation function of player i and $G_k^{v_i}$ is as defined in (6.4). By definition, $G_k^{v_i}(x)$ is the outcome of querying the bounded-lottery-value oracle of v_i with bound k and marginals x/k. Therefore, we can evaluate $f(x)$ using n bounded-lottery-value queries, one for each player. It remains to show that we can also evaluate the (multi-variate) derivative $\bigtriangledown f(x)$ of $f(x)$. Using definition (6.4) and Claim 6.5.3,

we take the partial derivative of $G_k^{v_i}$ with respect to x_j and simplify the resulting expression.

$$\begin{aligned}
\frac{\partial G_k^{v_i}}{\partial x_j}(x) &= \sum_{S\subseteq[m]} -1^{|S|} v_i(S) \sum_{R\subseteq S\setminus\{j\}} -1^{|R|+1} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-1} \\
&= \sum_{S\subseteq[m]\setminus\{j\}} -1^{|S|} \left(v_i(S\cup\{j\}) - v_i(S)\right) \sum_{R\subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-1} \\
&= \sum_{S\subseteq[m]} -1^{|S|} \left(v_i(S\cup\{j\}) - v_i(S)\right) \sum_{R\subseteq S} -1^{|R|} \left(1 - \frac{x_{\overline{R}}}{k}\right)^{k-1} \\
&= \sum_{S\subseteq[m]} v_i(S\cup\{j\}) \,\mathbf{Pr}\left[r_{k-1}\left(\frac{k-1}{k}x\right) = S\right] \\
&\quad - \sum_{S\subseteq[m]} v_i(S) \,\mathbf{Pr}\left[r_{k-1}\left(\frac{k-1}{k}x\right) = S\right]. \qquad \text{(B.3)}
\end{aligned}$$

The second equality follows by grouping the terms of the summation by the projection of S onto $[m]\setminus\{j\}$. The third equality follows from the observation that $v(S\cup\{j\}) - v(S) = 0$ when S includes j. The fourth equality follows by a simple re-arrangement and application of Claim 6.5.3.

Inspect the final form (B.3) in light of the definition of bounded-lottery-value oracles (Definition 6.2.1) and the definition of r_k (Section 6.4). Notice that the first term is the expected value of v_i over the $(k-1)$-bounded-lottery with marginals $\frac{k-1}{k}x$ and promise $\{j\}$. The second term is the expected value of v_i over the same lottery without the promise. Therefore, we can evaluate $\frac{\partial G_k^{v_i}}{\partial x_j}(x)$ using two queries to the bounded-lottery-value oracle of player i. This completes the proof of Claim B.2.1.

B.2.2 The Well-Conditioned Case

In this section, we make the following simplifying assumption: The objective function $f(x)$ of convex program (6.2), when restricted to any line in the feasible set $\mathcal{P}$, has a second derivative of magnitude at least $\lambda = \frac{\sum_{i=1}^n v_i([m])}{2^{\mathrm{poly}(n,m)}}$ everywhere, where the

polynomial in the denominator may be arbitrary. This is equivalent to requiring that every eigenvalue of the Hessian matrix of $f(x)$ has magnitude at least λ when evaluated at any point in $\mathcal{P}$. Under this assumption, we prove Lemma B.2.2.

Lemma B.2.2. *Assume the magnitude of the second derivative of $f(x)$ is at least $\lambda = \frac{\sum_{i=1}^{n} v_i([m])}{2^{\mathrm{poly}(n,m)}}$ everywhere. Algorithm 4.1, instantiated with $r = r_k$, can be simulated in time polynomial in n and m in expectation.*

Let x^* be the optimal solution to convex program (6.2). Algorithm 4.1 outputs a set of projects distributed as $r_k(x^*)$. The k-bounded-lottery rounding scheme, as described in Algorithm 6.1, requires making k independent decisions: for $\ell \in \{1, \ldots, k\}$, we draw p_ℓ uniformly from $[0, 1]$ and decide which interval I_j, if any, p_ℓ falls into. In other words, we find the minimum index j_ℓ (if any) such that $\sum_{j \le j_\ell} x^*_j/k \ge p_\ell$. Fix ℓ. For most realizations of p_ℓ, we can calculate j_ℓ using only coarse estimates $\widetilde{x}_j$ to x^*_j. Assume we have an *estimation oracle* for x^* that, on input δ, returns a δ-*estimate* $\widetilde{x}$ of x^*: Specifically, $\widetilde{x}_j - x^*_j \le \delta$ for each $j \in [m]$. If p_ℓ falls outside the "uncertainty zones" of $\widetilde{x}$, such as when $|p_\ell - \sum_{j' \le j} \widetilde{x}_{j'}/k| > \delta m/k$ for each $j \in [m]$, it is easy to see that we can correctly determine j_ℓ by using $\widetilde{x}$ in lieu of x. The total measure of the uncertainty zones of $\widetilde{x}$ is at most $2m^2\delta$, therefore p_ℓ lands outside the uncertainty zones with probability at least $1 - 2m^2\delta$. The following claim shows that if the estimation oracle for x^* can be implemented in time polynomial in $\log(1/\delta)$, then we can simulate the k-bounded-lottery rounding procedure in expected polynomial time.

Claim B.2.3. *Let x^* be the optimal solution of convex program* (6.2)*. Assume access to a subroutine $B(\delta)$ that returns a δ-estimate of x^* in* $\mathrm{poly}(n, m, \log(1/\delta))$ *time. Algorithm 4.1, instantiated with $r = r_k$, can be simulated in expected* $\mathrm{poly}(n, m)$ *time.*

Proof. Fix $\ell \in \{1, \ldots, k\}$. Draw $p_\ell \in [0, 1]$ uniformly at random as in the k-bounded-lottery rounding scheme in Algorithm 6.1. We will show how to find, in expected $\mathrm{poly}(n, m)$ time, the minimum index j_ℓ (if any) such that $\sum_{j \le j_\ell} x^*_j/k \ge p_\ell$.

The algorithm proceeds as follows: Start with $\delta = \delta_0 = \frac{1}{2m^2}$. Let $\widetilde{x} = B(\delta)$. While $|p_\ell - \sum_{j' \le j} \widetilde{x}_{j'}/k| \le \delta m/k$ for some $j \in [m]$ (i.e. p_ℓ may fall inside an "uncertainty

zone") do the following: let $\delta = \delta/2$, $\widetilde{x} = B(\delta)$ and repeat. After the loop terminates, we have a sufficiently accurate estimate of x^* to calculate j_ℓ.

It is easy to see that the above procedure is a faithful simulation of Algorithm (6.1) on x^*. It remains to bound its expected running time. Let $\delta_t = \frac{1}{2^{t+1}m^2}$ denote the value of δ at iteration t. By our initial assumption, iteration t takes time polynomial in n, m, and $\log(1/\delta_t)$, which by definition of δ_t is polynomial in n, m, and t. The probability this procedure does not terminate after t iterations is at most $2m^2\delta_t = 1/2^t$. Taken together, these two facts and a simple geometric summation imply that the expected runtime is polynomial in n and m. □

It remains to show that the estimation oracle $B(\delta)$ can be implemented in time polynomial in n, m and $\log(1/\delta)$. At first blush, one may expect that the ellipsoid method can be used in the usual manner here. However, there is one complication: we require an estimate $\widetilde{x}$ that is close to x^* *in solution space* rather than in terms of objective value. Using our assumption on the curvature of $f(x)$, we will reduce finding a δ-estimate of x^* to finding an $1-\epsilon(\delta)$ approximate solution to convex program (6.2). The dependence of ϵ on δ will be such that $\epsilon \geq \text{poly}(\delta)/2^{\text{poly}(n,m)}$, thereby we can invoke Claim B.2.1 to deduce that $B(\delta)$ can be implemented in $\text{poly}(n, m, \log(1/\delta))$ time.

Let $\epsilon = \epsilon(\delta) = \frac{\delta^2\lambda}{2\sum_i v_i([m])}$. Plugging in the definition of λ, we deduce that $\epsilon \geq \delta^2/2^{\text{poly}(n,m)}$, which is the desired dependence. It remains to show that if $\widetilde{x}$ is $(1-\epsilon)$-approximate solution to (6.2), then $\widetilde{x}$ is also a δ-estimate of x^*.

Using the fact that $f(x)$ is concave, and moreover its second derivative has magnitude at least λ, it a simple exercise to bound distance of any point x from the optimal point x^* in terms of its sub-optimality $f(x^*) - f(x)$, as follows:

$$f(x^*) - f(x) \geq \frac{\lambda}{2}||x - x^*||^2. \tag{B.4}$$

Assume $\widetilde{x}$ is a $(1-\epsilon)$-approximate solution to (6.2). Equation (B.4) implies that

$$||\widetilde{x} - x^*||^2 \leq \frac{2}{\lambda}\epsilon f(x^*) = \frac{\delta^2}{\sum_i v_i([m])} f(x^*) \leq \delta^2,$$

where the last inequality follows from the fact that $\sum_i v_i([m])$ is an upper-bound on the optimal value $f(x^*)$. Therefore, $||x - x^*|| \leq \delta$, as needed. This completes the proof of Lemma B.2.2.

B.2.3 Guaranteeing Good Conditioning

In this section, we propose a modification r_k^+ of the k-bounded-lottery rounding scheme r_k. We will argue that r_k^+ satisfies all the properties of r_k established so far, with one exception: the approximation guarantee of Lemma 6.4.2 is reduced to $1 - 1/e - 2^{-2mn}$. Then we will show that r_k^+ satisfies the curvature assumption of Lemma B.2.2, demonstrating that said assumption may be removed. Therefore Algorithm 4.1, instantiated with $r = r_k^+$ for combinatorial public projects with MRS valuations in the bounded-lottery-value oracle model, is $(1 - 1/e - 2^{-2mn})$ approximate and can be implemented in expected $\text{poly}(n, m)$ time. Finally, we show in Remark B.2.4 how to recover the 2^{-2mn} term to get a clean $1 - 1/e$ approximation ratio, as claimed in Theorem 6.3.1.

Let $\mu = 2^{-2nm}$. We define r_k^+ in Algorithm B.2. Intuitively, r_k^+ first chooses a tentative set $S \subseteq [m]$ of projects using r_k. Then it cancels its choice with small probability μ. Finally, with probability β it chooses a random project $j^* \in [m]$ and lets $S = \{j^*\}$. β is defined as the fraction of projects included in the original tentative choice of S. The motivation behind this seemingly bizarre definition of r_k^+ is purely technical: as we will see, it can be thought of as adding "concave noise" to r_k.

We can write the expected welfare $\mathbf{E}_{S \sim r_k^+(x)}[\sum_i v_i(S)]$ as follows.

$$\mathop{\mathbf{E}}_{S \sim r_k(x)} \left[(1 - \mu) \sum_i v_i(S) + \mu\beta \sum_i v_i(j^*) \right].$$

Using linearity of expectations and the fact that β is independent of the choice of j^* to simplify the expression, we get that $\mathbf{E}_{S \sim r_k^+(x)}[\sum_i v_i(S)]$ is equal to

$$(1 - \mu) \mathop{\mathbf{E}}_{S \sim r_k(x)} \left[\sum_i v_i(S) \right] + \mu \, \mathbf{E}[\beta] \frac{\sum_{j=1}^m \sum_{i=1}^n v_i(\{j\})}{m}.$$

Input: Fractional solution $x \in \mathbb{R}^m$ with $\sum_j x_j \leq k$, and $0 \leq x_j \leq 1$ for all j.
Output: Feasible solution $S \subseteq [m]$ with $|S| \leq k$

Let $S = r_k(x)$
Let $\beta = \frac{|S|}{m}$
Draw $q_1 \in [0,1]$ uniformly
if $q_1 \in [0, \mu]$ **then**
    Let $S = \emptyset$
    Draw $q_2 \in [0,1]$ uniformly
    if $q_2 \in [0, \beta]$ **then**
        Choose project $j^* \in [m]$ uniformly at random.
        Let $S = \{j^*\}$
    end if
end if

Algorithm B.2: The modified k-bounded-lottery rounding scheme r_k^+.

Observe that r_k chooses a project j with probability $1 - (1 - x_j/k)^k$. Therefore, the expectation of β is $\frac{\sum_j 1-(1-x_j/k)^k}{m}$. This gives:

$$\mathop{\mathbf{E}}_{S \sim r_k^+(x)} \left[\sum_i v_i(S) \right] = (1-\mu) \mathop{\mathbf{E}}_{S \sim r_k(x)} \left[\sum_i v_i(S) \right] + \frac{\mu}{m^2} \left(\sum_{j=1}^{m} \sum_{i=1}^{n} v_i(\{j\}) \right) \left(\sum_{j=1}^{m} 1 - (1 - x_j/k)^k \right). \tag{B.5}$$

It is clear that the expected welfare when using $r = r_k^+$ is within $1 - \mu = 1 - 2^{-2nm}$ of the expected welfare when using $r = r_k$ in the instantiation of Algorithm 4.1. Using Lemma 6.4.2, we conclude that r_k^+ is a $(1 - 1/e - 2^{-2nm})$-approximate rounding scheme. Moreover, using Lemma 6.4.1, as well as the fact that $1 - (1 - x_j/k)^k$ is a concave function, we conclude that r_k^+ is a convex rounding scheme. Therefore, this establishes the analogues of Lemmas 6.4.2 and 6.4.1 for r_k^+. It is elementary to verify that our proof of Lemma B.2.2 can be adapted to r_k^+ as well.

It remains to show that r_k^+ is "sufficiently concave". This would establish that the conditioning assumption of Section B.2.2 is unnecessary for r_k^+. We will show

that expression (B.5) is a concave function with curvature of magnitude at least $\lambda = \frac{\sum_{i=1}^{n} v_i([m])}{em^2 2^{2nm}}$ everywhere. Since the curvature of concave functions is always non-positive, and moreover the curvature of the sum of two functions is the sum of their curvatures, it suffices to show that the second term of the sum (B.5) has curvature of magnitude at least λ. We note that the curvature of $\sum_j \left(1 - (1 - x_j/k)^k\right)$ is at least e^{-1} over $x \in [0,1]^m$. Therefore, the curvature of the second term of (B.5) is at least

$$\frac{\mu}{m^2}\left(\sum_i v_i([m])\right) e^{-1} = \lambda$$

as needed.

Remark B.2.4. *In this section, we sacrificed 2^{-2nm} in the approximation ratio in order to guarantee expected polynomial runtime of our algorithm even when convex program* (6.2) *is not well-conditioned. This loss can be recovered to get a clean $1-1/e$ approximation. The argument is identical to that employed for combinatorial auctions in Remark B.1.4, and is omitted.*

Bibliography

[1] N. Andelman, Y. Azar, and M. Sorani. Truthful approximation mechanisms for scheduling selfish related machines. *Theory of Computing Systems*, 40(4):423–436, 2007.

[2] A. Archer. *Mechanisms for Discrete Optimization with Rational Agents*. PhD thesis, Cornell University, 2004.

[3] A. Archer and É. Tardos. Truthful mechanisms for one-parameter agents. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 482–491, 2001.

[4] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[5] L. Ausubel and P. Milgrom. The lovely but lonely Vickrey auction. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*. MIT Press, 2006.

[6] Y. Azar and L. Epstein. Approximation schemes for covering and scheduling on related machines. In *Proceedings of the First International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, volume 1444 of *Lecture Notes in Computer Science*, pages 39–47, 1998.

[7] M. Babaioff, R. Lavi, and E. Pavlov. Single-value combinatorial auctions and algorithmic implementation in undominated strategies. *Journal of the ACM (JACM)*, 56(1):1–32, 2009.

[8] X. Bei and Z. Huang. Towards optimal Bayesian algorithmic mechanism design. In *Proceedings of the 22nd ACM Symposium on Discrete Algorithms (SODA)*, pages 720–733, 2011.

[9] R. Beier and B. Vöcking. Typical properties of winners and losers in discrete optimization. *SIAM Journal on Computing*, 35(4):855–881, 2006.

[10] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. SIAM, 2001.

[11] S. Bikhchandani, S. Chatterji, R. Lavi, A. Mu'alem, N. Nisan, and A. Sen. Weak monotonicity characterizes deterministic dominant-strategy implementation. *Econometrica*, 74(4):1109–1132, 2006.

[12] L. Blumrosen and N. Nisan. On the computational power of iterative auctions. In *Proceedings of the 6th ACM Conference on Electronic Commerce (EC)*, pages 29–43, 2005.

[13] L. Blumrosen and N. Nisan. Combinatorial auctions (a survey). In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, 2007.

[14] L. Blumrosen and N. Nisan. Informational limitations of ascending combinatorial auctions. *Journal of Economic Theory*, 145(3):1203–1223, 2010.

[15] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[16] P. Briest, P. Krysta, and B. Vöcking. Approximation techniques for utilitarian mechanism design. In *Proceedings of the 36th ACM Symposium on Theory of Computing (STOC)*, pages 39–48, 2005.

[17] D. Buchfuhrer, S. Dughmi, H. Fu, R. Kleinberg, E. Mossel, C. Papadimitriou, M. Schapira, Y. Singer, and C. Umans. Inapproximability for VCG-based combinatorial auctions. In *Proceedings of the 21st ACM Symposium on Discrete Algorithms (SODA)*, pages 518–536, 2010.

[18] D. Buchfuhrer, M. Schapira, and Y. Singer. Computation and incentives in combinatorial public projects. In *Proceedings of the 11th ACM Conference on Electronic Commerce (EC)*, pages 33–42, 2010.

[19] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint. In *Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 182–196, 2007.

[20] G. Christodoulou and A. Kovács. A deterministic truthful PTAS for scheduling related machines. In *Proceedings of the 21st ACM Symposium on Discrete Algorithms (SODA)*, pages 1005–1016, 2010.

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

[22] V. Conitzer and T. Sandholm. Computational criticisms of the revelation principle. In *Proceedings of the 5th ACM Conference on Electronic Commerce (EC)*, pages 262–263, 2004.

[23] P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press., 2006.

[24] P. Dhangwatnotai, S. Dobzinski, S. Dughmi, and T. Roughgarden. Truthful approximation schemes for single-parameter agents. In *Proceedings of the 49th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 15–24, 2008.

[25] S. Dobzinski. Two randomized mechanisms for combinatorial auctions. *Proceedings of the 10th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 89–103, 2007.

[26] S. Dobzinski. An impossibility result for truthful combinatorial auctions with submodular valuations. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 139–148, 2011.

[27] S. Dobzinski and S. Dughmi. On the power of randomization in algorithmic mechanism design. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 505–514, 2009.

[28] S. Dobzinski, H. Fu, and R. Kleinberg. Truthfulness via proxies. *arXiv:1011.3232*, 2010.

[29] S. Dobzinski and N. Nisan. Limitations of VCG-based mechanisms. In *Proceedings of the 38th ACM Symposium on Theory of Computing (STOC)*, pages 338–344, 2007.

[30] S. Dobzinski and N. Nisan. Mechanisms for multi-unit auctions. In *Proceedings of the 8th ACM Conference on Electronic Commerce (EC)*, pages 346–351, 2007.

[31] S. Dobzinski and N. Nisan. Multi-unit auctions: beyond Roberts. In *Proceedings of the 12th ACM Conference on Electronic Commerce (EC)*, pages 233–242, 2011.

[32] S. Dobzinski, N. Nisan, and M. Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proceedings of the 36th ACM Symposium on Theory of Computing (STOC)*, pages 610–618, 2005.

[33] S. Dobzinski, N. Nisan, and M. Schapira. Truthful randomized mechanisms for combinatorial auctions. In *Proceedings of the 37th ACM Symposium on Theory of Computing (STOC)*, pages 644–652, 2006.

[34] S. Dughmi. A truthful randomized mechanism for combinatorial public projects via convex optimization. In *Proceedings of the 12th ACM Conference on Electronic Commerce (EC)*, pages 263–272, 2011.

[35] S. Dughmi and T. Roughgarden. Black-box randomized reductions in algorithmic mechanism design. In *Proceedings of the 51st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 775–784, 2010.

[36] S. Dughmi, T. Roughgarden, and M. Sundararajan. Revenue submodularity. In *Proceedings of the 10th ACM Conference on Electronic Commerce (EC)*, pages 243–252, 2009.

[37] S. Dughmi, T. Roughgarden, and Q. Yan. From convex optimization to randomized mechanisms: toward optimal combinatorial auctions. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 149–158, 2011.

[38] S. Dughmi and J. Vondrák. Limitations of randomized mechanisms for combinatorial auctions. In *Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science (FOCS)*, 2011. To appear.

[39] L. Epstein and J. Sgall. Approximation schemes for scheduling on uniformly related and identical parallel machines. *Algorithmica*, 39(1):43–57, 2004.

[40] L. Epstein and R. van Stee. Maximizing the minimum load for selfish agents. In *Proceedings of the 8th Conference on Latin American Theoretical Informatics (LATIN)*, volume 4957 of *Lecture Notes in Computer Science*, pages 264–275, 2008.

[41] U. Feige. On maximizing welfare where the utility functions are subadditive. In *Proceedings of the 37th ACM Symposium on Theory of Computing (STOC)*, pages 122–142, 2006.

[42] U. Feige, V. S. Mirrokni, and J. Vondrák. Maximizing non-monotone submodular functions. In *Proceedings of the 48th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 461–471, 2007.

[43] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[44] G. Gens and E. Levner. Approximation algorithms for certain universal problems in scheduling theory. *Soviet Journal of Computers and System Sciences*, 6:31–36, 1978.

[45] G. V. Gens and E. V. Levner. Fast approximation algorithms for knapsack type problems. In *Optimization Techniques*, pages 185–194, 1980.

[46] T. Groves. Incentives in teams. *Econometrica*, 41(4):617–31, 1973.

[47] J. Hartline, R. Kleinberg, and A. Malekian. Multi-parameter Bayesian algorithmic mechanism design. In *Proceedings of the 22nd ACM Symposium on Discrete Algorithms (SODA)*, pages 734–747, 2011.

[48] J. D. Hartline and B. Lucier. Bayesian algorithmic mechanism design. In *Proceedings of the 41st ACM Symposium on Theory of Computing (STOC)*, pages 301–310, 2010.

[49] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, 1992.

[50] D. Hochbaum and D. B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.

[51] R. Holzman, N. Kfir-Dahav, D. Monderer, and M. Tennenholtz. Bundling equilibrium in combinatorial auctions. *Games and Economic Behavior*, 47(1):104–123, 2004.

[52] D. S. Johnson and K. A. Niemi. On knapsacks, partitions, and a new dynamic programming technique for trees. *Mathematics of Operations Research*, 8(1):1–14, 1983.

[53] J. A. Kelner and D. A. Spielman. A randomized polynomial-time simplex algorithm for linear programming. In *Proceedings of the 37th ACM Symposium on Theory of Computing (STOC)*, pages 51–60, 2006.

[54] S. Khot, R. J. Lipton, E. Markakis, and A. Mehta. Inapproximability results for combinatorial auctions with submodular utility functions. *Algorithmica*, 52(1):3–18, 2008.

[55] P. Klemperer. *Auctions: Theory and Practice*. Princeton University Press, 2004.

[56] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 2008.

[57] V. Krishna. *Auction Theory*. Academic Press, 2002.

[58] R. Lavi. Computationally efficient approximation mechanisms. In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, chapter 12, pages 301–329. Cambridge University Press, 2007.

[59] R. Lavi, A. Mu'alem, and N. Nisan. Towards a characterization of truthful combinatorial auctions. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 574–583, 2003.

[60] R. Lavi and C. Swamy. Truthful and near-optimal mechanism design via linear programming. In *Proceedings of the 46th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 595–604, 2005.

[61] D. Lehmann, L. Oćallaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.

[62] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt. Bicriteria network design problems,. *Journal of Algorithms*, 28(1):142 – 171, 1998.

[63] A. Mas-Colell, W. Whinston, and J. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[64] E. Maskin and J. Riley. Optimal auctions with risk averse buyers. *Econometrica*, 52(6):1473–1518, 1984.

[65] P. Milgrom. *Putting Auction Theory to Work*. Cambridge University Press, 2004.

[66] J. Mirrlees. An exploration in the theory of optimum income taxation. *The Review of Economic Studies*, 38(2):175–208, 1971.

[67] A. Mu'alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *Proceedings of the Eighteenth national conference on Artificial intelligence (AAAI)*, pages 379–384, 2002.

[68] R. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

[69] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions – I. *Mathematical Programming*, 14(3):265–294, 1978.

[70] N. Nisan. Introduction to mechanism design (for computer scientists). In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, 2007.

[71] N. Nisan and A. Ronen. Computationally feasible VCG-based mechanisms. In *Proceedings of the 2nd ACM Conference on Electronic Commerce (EC)*, pages 242–252, 2000.

[72] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behaviour*, 35:166 – 196, 2001.

[73] J. G. Oxley. *Matroid Theory*. Oxford University Press, 1992.

[74] C. Papadimitriou, M. Schapira, and Y. Singer. On the hardness of being truthful. In *Proceedings of the 49th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 250–259, 2008.

[75] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th ACM Symposium on Theory of Computing (STOC)*, pages 475–484, 1997.

[76] J. Riley and W. Samuelson. Optimal auctions. *American Economic Review*, 71(3):381–392, 1981.

[77] K. Roberts. The characterization of implementable choice rules. In J.-J. Laffont, editor, *Aggregation and Revelation of Preferences. Papers presented at the first European Summer Workshop of the Economic Society*, pages 321–349. North-Holland, 1979.

[78] J.-C. Rochet. A necessary and sufficient condition for rationalizability in a quasi-linear context. *Journal of Mathematical Economics*, 16(2):191 – 200, 1987.

[79] H. Röglin and S.-H. Teng. Smoothed analysis of multiobjective optimization. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 681–690, 2009.

[80] S. Sahni. General techniques for combinatorial approximation. *Operations Research*, 25(6):920–936, 1977.

[81] M. E. Saks and L. Yu. Weak monotonicity suffices for truthfulness on convex domains. In *Proceedings of the 6th ACM Conference on Electronic Commerce (EC)*, pages 286–293, 2005.

[82] A. Spence. Competitive and optimal responses to signals: An analysis of efficiency and distribution. *Journal of Economic Theory*, 7(3):296–332, 1974.

[83] D. A. Spielman and S.-H. Teng. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. In *Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC)*, pages 296–305, 2001.

[84] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.

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

[86] B. Vöcking. A universally-truthful approximation scheme for multi-unit auctions. Manuscript, 2011.

[87] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 39th ACM Symposium on Theory of Computing (STOC)*, pages 67–74, 2008.