MECHANISMS FOR DISCRETE OPTIMIZATION WITH RATIONAL AGENTS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Aaron Francis Archer

January 2004

MECHANISMS FOR DISCRETE OPTIMIZATION WITH RATIONAL AGENTS

Aaron Francis Archer, Ph.D.

Cornell University 2004

We study "discrete optimization with a blindfold." We are given an optimization problem such as finding the shortest path in a graph or balancing load on machines, but some of the relevant data, such as edge costs or machine speeds, is missing. There are various selfish economic agents who each know part of the data, and who are affected by the decisions we make. We can ask them to report the data, but they might lie if it is in their self-interest to do so. We aim to design truthful mechanisms: algorithms for solving our optimization problems that, with the help of monetary incentives, induce the agents to report the true data.

We identify and address four major problems with the Vickrey-Clarke-Groves (VCG) mechanism, the most famous general technique for designing truthful mechanisms. First, the VCG mechanism can be used only if we want to maximize the overall social welfare, e.g., minimize the cost to the agents. We develop other techniques for optimizing other types of objective functions. Second, even if our goal is to maximize the overall social welfare, the VCG mechanism may be computationally intractable because the underlying optimization problem is NP-hard. We investigate how to develop approximation algorithms that preserve the truth-inducing properties of the VCG mechanism. Third, even though the VCG mechanism can be used to minimize the total cost incurred by the agents, the amount

the mechanism must pay to the agents can be significantly higher. We prove that in some natural cases, this is an inherent difficulty with truthful mechanisms, and is not peculiar to the VCG mechanism. Fourth, even though it is never in an agent's individual selfish interest to lie to the VCG mechanism, coalitions of agents can sometimes benefit by lying in a coordinated way. We study what types of coordination must occur for this type of cheating to be successful.

BIOGRAPHICAL SKETCH

Aaron Archer grew up in Tucson, Arizona, where he graduated from University High School in 1994. He earned his B.S. in Mathematics from Harvey Mudd College in spring 1998, his M.S. in Operations Research from Cornell University in fall 2001, and expects to receive his Ph.D. in Operations Research from Cornell in January 2004. Other influential educational experiences include the Hampshire College Summer Studies in Mathematics, where he has been both a student and instructor, Joe Gallian's mathematics REU program at the University of Minnesota, Duluth, and the Budapest Semesters in Mathematics program.

ACKNOWLEDGEMENTS

I would like to express heartfelt thanks to the many who have helped me through graduate school. First and foremost, I thank my advisor, Éva Tardos. Her sound advice, patience, generosity and encouragement have been invaluable to me throughout my career at Cornell. I also cannot imagine having spent the last few years without my regular conversations with David Shmoys. One of the best communicators I know, I have found him to be an outstanding research collaborator, teaching mentor, and source of personal advice. I thank these two, Jon Kleinberg, and Lou Billera for serving on my thesis committee.

While visiting UC Berkeley for a semester, I had the good fortune of sharing an office with Anupam Gupta. I have David Williamson, Sridhar Rajagopalan, and Ron Fagin to thank for two summers of academic and personal growth while visiting IBM Almaden.

I would like to thank my mentors from my undergraduate years at Harvey Mudd College, particularly Art Benjamin, Francis Su, Bob Borrelli and Mike Moody, for shaping me as a mathematician and keeping up with me over the years. Also Joe Gallian for the opportunity to spend many excellent weeks doing combinatorics in Duluth, David Kelly for running the fantastic Hampshire College summer program that convinced me to pursue mathematics, and Rennie Mirollo for instructing me that "the blackboard starts in the upper left hand corner" and imparting other valuable wisdom early in my teaching career.

My years at Cornell would have been much less enjoyable without the friendship and good wit of my housemates, Rif, Mohan, Ranjith, Tom and Peter. Laughter and many great meals came out of our kitchen. Thanks to Nathan and Millie for many trips to Wegmans, without which I would surely have starved. Thanks to

my Ultimate frisbee teammates on Vitamin I for providing me with a valuable athletic and social outlet.

Most of all, thank you to my loving family, for their constant support and for making me who I am, and to Melissa for continually grounding and refreshing my spirit.

I thank the Fannie and John Hertz Foundation for their generous support of my graduate studies. My work has also been supported by NSF grant CCR-0113371.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

Traditional algorithms design typically assumes that all components of the system under study are working together. In situations where the various components are owned by different economic entities, it makes more sense to model these components as *rational* (or *selfish*) *agents*. A selfish agent will deviate from the protocol if she thinks it is in her self-interest to do so. This motivates us to attempt to create protocols that are *strategyproof*, which means that it is in each agent's self-interest to adhere to them.

Game theory is a branch of mathematics that attempts to understand how rational agents will behave in strategic situations. It has long been widely applied in economics, where it is a standard part of most curricula. One branch of game theory called *mechanism design* concerns the construction of *mechanisms*, games whose purpose is to get the agents to behave in a certain way so as to achieve some desired outcome. Historically, this field grew out of social choice theory, which typically dealt with things like voting systems, taxation policy and the allocation of public goods. Now that mechanism design is being applied to algorithms research, new issues such as computational efficiency arise. To distinguish this new setting, it is often called *algorithmic mechanism design*, which is the subject of this dissertation.

Interest in algorithmic mechanism design has come from both the algorithms side and the economics side. Influential algorithms papers by Nisan and Ronen [58] and Feigenbaum, et al. [22] promote mechanism design as a means for creating ef-

fective protocols for the Internet, since different pieces of the Internet are owned, operated and used by different economic entities who might attempt to gain by manipulating the protocols. At the same time, economists have shown new interest in computation, motivated by the rise of electronic commerce, Internet auction sites like eBay, and recent large-scale auctions by the United States and other governments to allocate spectrum rights to mobile phone providers. In particular, there has been a desire to use *combinatorial auctions*, which are auctions that allow participants to place bids on combinations of goods rather than single goods. While there are some mechanisms for these types of auctions that have many desirable game-theoretic properties, they represent computational difficulties, hence the interest in algorithms research from economists.

The angle we take is that we want to do "discrete optimization with a blindfold." There is some discrete optimization problem that we want to solve, but we do not know all of the data. Instead, there are various rational agents who each know some of the data. We can ask them to report it, but they might lie. Our task is to design mechanisms involving payments that induce the agents to tell the truth, so that we are performing our computations on the correct data, and at the same time optimize the objective function we are interested in. We will have to deal with both the game theoretic issues of getting the agents to tell the truth, i.e., designing *truthful* mechanisms, and the algorithmic issues of efficiently computing the outcomes of these mechanisms.

The most famous positive result in mechanism design is the Vickrey-Clarke-Groves (VCG) mechanism [69, 14, 31], a general technique for creating truthful mechanisms. This mechanism is widely touted in the literature, but it can sometimes have some serious drawbacks, both in terms of the results it gives and the

difficulty in computing it. The focus of this dissertation is to address these problems.

1.2 Example problems

In this section we describe two mechanism design problems representative of some of the topics we will address in this work. First we discuss the Vickrey "second-price" auction, since it occupies an important place in the history of mechanism design, and has made a significant impact in practice. Then we discuss the shortest path problem, since it is one of the problems we will study and it shares the discrete optimization flavor of the rest of this dissertation.

1.2.1 A success story: the Vickrey "second-price" auction

The *second-price* auction is an innovative auction form. There is one item for sale, and multiple bidders, who each submit a sealed bid in an envelope. The auctioneer then opens all the envelopes and awards the item to the highest bidder. However, in contrast to the first-price auction where the bidder would pay what she bid, in this auction we charge the winner the amount of the second-highest bid. Under some simple assumptions about the motivations of the bidders, this auction is truthful. In this case, this means that each bidder's best strategy is to bid the highest amount she would be willing to pay for the object, no matter what the other bidders do. This derives from the fact that the winner cannot improve the price she pays by lowering her bid. We return to this auction in Section 2.1 for a rigorous proof. Viewed as an optimization problem, this auction solves the problem of awarding an object to the bidder who values it most.

The second-price auction works equally well for *procurement* auctions, where

the mechanism is acting as a buyer rather than a seller. For instance, the mechanism could be run by the government, where the agents are contractors bidding on a public project. The contract is awarded to the lowest bidder, who is paid the amount of the second-lowest bid.

The second-price auction is often called the Vickrey auction after William Vickrey, who popularized and generalized it, and studied its theoretical properties in his landmark 1961 paper [69]. However, some scholars trace the history of this auction back to the famous German poet Johann Wolfgang von Goethe, who used it as early as 1797 in some innovative negotiations with publishers [50, 44].

The United States Treasury now uses a slight variant on the Vickrey auction for its competitive bidding process that determines the interest rates for treasury securities purchased in large blocks by financial institutions. Instead of all winning bidders paying what they bid, they all pay the bid of the highest losing bidder. As early as 1959, Milton Friedman advocated this in front of Congress. The Treasury experimented with it briefly in the 1970's, before returning to the "pay what you bid" rule for the auction winners. However, a 1991 scandal in which Salomon Brothers, Inc. admitted to submitting fraudulent bids (which resulted in a fine of $290 millon) and two other successful attempts to cheat the bidding system prompted Friedman and other economists to advocate the Vickrey auction again [63, 24, 12]. The Treasury experimented with selling 2-year and 5-year notes using the Vickrey auction, starting in 1992. This trial was so successful that they switched all treasury issues to the Vickrey format in 1998. According to their report [43], the switch resulted in a more equitable spread of winners, and somewhat decreased costs for the American taxpayer. They attributed this to more aggressive bidding behavior by a wider range of institutions, who no longer had nearly

as much to lose if they slightly misjudged the value they should bid. The popular online auction site eBay also uses a variant on the Vickrey auction.[1]

The Vickrey auction is an example where the study of truthful mechanism design has had a significant positive impact on policy and practice. In recognition of this, Vickrey received the 1996 Nobel Prize in economics for his work.

1.2.2 Shortest paths

Suppose we want to send a package (or data packet) through a transportation (or data) network from a source node to a destination node. We model the network as a graph, where the edges of the graph denote the transportation links of the network. If there is a cost associated with shipping on each edge that we use, then we want to choose the cheapest path. This is the classic shortest path problem. The cost is usually viewed as a type of distance, hence the name "shortest" path.

Now suppose that each edge is owned by a different economic agent who will incur her edge cost if her edge is used. In order to get an agent to let us use her edge, we will have to pay her to reimburse her. The trick is, only that agent knows her true cost. Can we still manage to choose the cheapest path?

Suppose that we use the following naive protocol for selecting our path. We ask each agent to report the cost for her edge, and we select the shortest path with respect to these costs. If an agent reports her true edge cost and ends up being on the selected path, then she obtains a profit of zero, since the payment she receives equals the cost she incurs. Such an agent should overstate her cost somewhat, so that if she is on the selected path she will be paid more than the cost she incurs, obtaining a positive profit. However, if she exaggerates too much then some other

[1] One key difference is that people are allowed to submit multiple bids over time.

path will become less expensive and our protocol will choose that path instead. Thus, it might be difficult for her to figure out how much she should overstate her cost, since this depends on what costs the other agents decide to report. In any event, we can expect the agents to lie about their costs, but it is unpredictable how much they will do so.

Nisan and Ronen applied the VCG result to create a truthful mechanism for this problem [58]. The protocol still asks each edge to report her cost, and chooses the shortest path with respect to these costs. However, each edge that is on the selected path is paid *more* than the cost she announced. These payments are designed very cleverly so that each agent's best strategy is always to report her true cost, no matter what the other agents do. Since we assume the agents are rational, they will always play this best strategy. Thus, this mechanism succeeds in making the agents behave predictably, and it maximizes the "social welfare," meaning that it minimizes the total cost incurred by the agents.

Every truthful mechanism must pay a premium to the agents in order to induce truth-telling. The VCG mechanism is not the only truthful mechanism for this problem, so it may not be the one that results in the lowest overall payments. We advocate the study of *frugal* mechanisms (a term we coined in [5, 6]), which keep the payments low. This is the main topic of Chapter 3.

Feigenbaum et al. [21] studied the shortest path problem in a slightly different model, where the nodes incur costs rather than the edges. They proposed this as a model for the problem of routing data packets on the Internet through least-congested paths. The nodes represent *autonomous systems* (AS's), the various subnetworks of the Internet that are owned by different entities, such as the Sprint network, the AT&T network, or your local Internet service provider. Transit traffic

refers to packets that travel through an AS, but whose origin and destination lie outside the AS. When one AS hands off traffic to another, they exchange some congestion information in order to help route the traffic through uncongested AS's. Since accepting transit traffic costs the AS, it may benefit by exaggerating its claims about its own congestion. In using a mechanism with side payments to induce truth-telling, the hope is that we would improve the system performance by taking away the incentive for an AS to ever misreport its congestion.

1.3 Mathematical framework

In this dissertation, we adopt the algorithmic mechanism design framework set forth by Nisan and Ronen [58]. We describe all of the notation first, then show how the Vickrey auction and shortest path examples fit in.

There is a set of agents $\mathcal{N}$ that the mechanism must deal with, and a collection of outcomes $\mathcal{O}$ from which it wishes to choose. Each agent i is assumed to have a *type* t_i, which encapsulates all relevant information about that agent. Agent i's type is drawn from some set $\mathcal{T}_i$, which is public knowledge, but the actual value of t_i is known only to agent i. We will often refer to i's type as its *secret data* or *true value*. Let t denote the vector of types. The mechanism works by asking each agent to report her type, and computing an outcome and a set of payments based on the reported types. We refer to i's reported type as its *bid* b_i, and let b denote the vector of bids. We let $\mathcal{T} = \prod_{i \in \mathcal{N}} \mathcal{T}_i$, $P_i : \mathcal{T} \to \mathbb{R}$ be the payment to agent i, and $P = (P_i)_{i \in \mathcal{N}}$ be the payment *scheme*. Thus, a mechanism $\mathcal{M}$ consists of a pair (O, P), where $O : \mathcal{T} \to \mathcal{O}$ is the *output* function, and P is the payment scheme.

Each agent i has a *valuation* function $v_i : \mathcal{O} \to \mathbb{R}$ that describes her intrinsic preferences for the various outcomes. This valuation function is one of the items

specified by the type t_i, so technically it should be written as $v_i(\cdot, t_i)$, but we will usually write it as $v_i(\cdot)$, keeping in mind that it depends implicitly on t_i.

Each agent's goal is to maximize her *utility* function,[2] which we assume to be of the form $u_i(o, P_i) = v_i(o) + P_i$. In the literature, this form of utility function is called *quasi-linear*. Since the mechanism determines the outcome and payments from the bids, we will often abuse notation by writing $u_i(b)$ and $v_i(b)$ as shorthand for $u_i(O(b), P_i(b))$ and $v_i(O(b))$.

Of course, an agent's utility depends both on her bid and the other agents' bids. Let b_{-i} denote the vector of all bids other than agent i's. Similarly, let t_{-i} denote the vector of all other agents' types, and $\mathcal{T}_{-i} = \prod_{j \in \mathcal{N}-i} \mathcal{T}_j$ the space of those type vectors. For convenience, we often write b as (b_{-i}, b_i). We say that truth-telling is a *dominant strategy* for agent i if $u_i(b_{-i}, t_i) \geq u_i(b_{-i}, b_i)$ for all $b_{-i} \in \mathcal{T}_{-i}$ and $b_i, t_i \in \mathcal{T}_i$. That is, agent i's best strategy is to report her true type, no matter what the other agents do.

If there is a payment scheme P such that the mechanism (O, P) is truthful, then P is said to *implement* the output function O, and O is said to be *implementable*. Most of the time, the goal of our mechanisms will be to select $o \in \mathcal{O}$ to as to maximize or minimize some objective function $g(o, t)$. In other words, we want to implement the output function O that optimizes g, or at least one that approximately optimizes it.

Often, the only role of the type t_i is to specify the valuation function v_i. In those cases, we might as well simply declare the type to *be* the valuation function. However, the type might also contain some extra information that is relevant to the objective function $g(o, t)$ but not to the agent's valuation. For instance, in a

[2]We do *not* assume in general that these are von Neumann-Morgenstern utilities. See Section 2.5.

network flow problem where the agent owns an edge, her type might specify the capacity of that edge.

In the vast majority of recent work in algorithmic mechanism design, the outcomes and valuation and functions are assumed to have a very simple form. Each outcome is assumed to pick some set of "winners," and each agent's type is a real number describing her valuation for winning. Thus, if q_i is a binary (i.e., $\{0, 1\}$) variable indicating whether i wins, then $v_i = v_i(q_i) = \pm t_i q_i$, where the sign depends on whether the agent accrues a cost or benefit from winning. The vector $(q_i)_{i \in \mathcal{N}}$ is called the *allocation*. Notice that the general model would allow agent i's valuation to depend on *all* of the allocations q_j, $j \in \mathcal{N}$, but here it depends only on q_i. We use this binary allocation model for much of our work, but also consider a more general model where the q_i can be any non-negative real number. We refer to valuations of the form $v_i = \pm t_i q_i$ as *one-dimensional linear* valuations. All of our work assumes this form for the valuations.

Examples. We can describe the Vickrey auction using the binary allocation model. In order for this auction to fit in this model, we must assume that each agent cares only about whether or not she wins and how much she pays, not about who else might win, or what they pay. Agent i's type t_i is a non-negative real number describing her valuation for winning. Each possible outcome $o \in \mathcal{O}$ is described by a vector of indicator functions $(q_i)_{i \in \mathcal{N}}$, where $q_i(b) = 1$ if i wins, and otherwise $q_i(b) = 0$. Thus, agent i's valuation function is $v_i(b) = t_i q_i(b)$, and her utility is $u_i(b) = t_i q_i(b) + P_i(b)$. Notice that if i wins, P_i is negative, because the agent is paying money to the mechanism.

If b_i is the largest bid in b, then the Vickrey auction chooses $O(b)$ to be the allocation that gives the item to i. (If there is a tie for the highest bid, this

must be resolved by some tie-breaking rule, which we have left unspecified.) If i is the winner, then $P_j(b) = 0$ for $j \neq i$, and $P_i(b) = -\max_{j \neq i} b_j$. One way of looking at the Vickrey auction is that it maximizes the objective function $g(o, t) = \sum_{i \in \mathcal{N}} v_i(o)$. We will prove in Section 2.1 that the Vickrey auction is truthful.

In the shortest path problem, the agents are the edges. Agent i's type t_i is a non-negative number denoting the cost she incurs if her edge is used. The feasible outputs O are all allocation vectors $(q_i)_{i \in \mathcal{N}}$ where the winning edges form a path from the source to the destination in the graph. Thus, her valuation is $v_i(b) = -t_i q_i(b)$ and her utility is $u_i(b) = -t_i q_i(b) + P_i(b)$. Since the mechanism will pay money to the agents, the P_i's are positive.

1.4 Our contributions

As we mentioned in Section 1.1, the major positive result in mechanism design is the VCG mechanism. This is a truthful mechanism that handles arbitrary valuation functions for the agents. We describe the mechanism precisely in Section 2.3.2. For now, it is enough to know that the VCG mechanism selects the outcome that maximizes the total valuation of all the agents, which is called the *utilitarian* objective function.

Our contribution is to address four major drawbacks of the VCG mechanism. First, in a setting where the mechanism is buying something from the agents, the VCG mechanism can sometimes be forced to overpay by a large factor compared to the actual costs incurred by the agents. Second, the VCG mechanism works only to optimize the utilitarian objective function, whereas in many optimization problems we are interested in some other objective. Third, even in situations where the utilitarian objective is appropriate, the VCG mechanism is often NP-

hard to compute. Fourth, the VCG mechanism is resistant to manipulation by single agents, but sometimes coalitions of agents can collude to manipulate the mechanism. We devote one chapter to addressing each of these drawbacks.

Since VCG has these problems, we will be looking for alternate truthful mechanisms that solve them. Unfortunately, there is a theorem by Roberts (see Section 2.3.2) that says VCG is essentially the only truthful mechanism, if the agents' valuations are allowed to be arbitrary. Therefore, we must restrict the form of the agents' valuations in order to be able to create other truthful mechanisms. The reason that we consider only problems with one-dimensional linear valuations is that these admit a much wider class of truthful mechanisms.

In order for a mechanism to be useful, we must be able to compute the output and payments efficiently. Therefore, we will always focus on algorithms that run in polynomial time. Many of the objective functions that we consider are NP-hard to optimize exactly, so we will use approximation algorithms. A solution to an optimization problem is α-*approximate* if it is within a multiplicative factor of α of the optimum. An α-*approximation algorithm* is an algorithm that runs in polynomial time, and always returns a solution that is α-approximate.[3]

It is often difficult to design an approximation algorithm for the output function in a way that allows us to attach a payment scheme yielding a truthful mechanism. Overcoming this difficulty is one of the major themes of Chapters 4 and 5.

We now give an overview of the organization of this dissertation.

Chapter 2: Preliminaries We give a brief survey of the most relevant results in mechanism design. We also develop some tools for computing payments efficiently,

[3]See the book [68] for a good introduction to the field of approximation algorithms.

and discuss randomized mechanisms, which will be used in later chapters.

Chapter 3: Frugality We consider cases in which the mechanism is hiring a team of agents to perform a task, such as the shortest path problem described in Section 1.2.2. We investigate the "price of truthfulness" – how much a truthful mechanism can be forced to overpay, compared to the actual costs incurred by the agents. The VCG mechanism can be forced to overpay by large amounts, so we investigate other truthful mechanisms for shortest paths and related problems. Roughly, the results of this chapter show that the overpayment is inherent to the problem, not specific to VCG. This chapter is based primarily on the paper "Frugal path mechanisms" [6], co-authored with Éva Tardos.

Chapter 4: Truthful discrete optimization We design polynomial-time truthful mechanisms for a variety of optimization problems with non-utilitarian objectives, such as load balancing, maximum flow, and facility location. This chapter is based primarily on the paper "Truthful mechanisms for one-parameter agents" [5], co-authored with Éva Tardos.

Chapter 5: Randomized approximations to VCG We consider cases where we do want to maximize the utilitarian objective, but cannot use VCG because it is NP-hard to compute the optimal output. We show how to adapt the technique of randomized rounding to obtain truthful mechanisms that approximately optimize the objective. We describe this technique in the context of a particular combinatorial auction problem. This chapter is based on the paper "An approximate truthful mechanism for combinatorial auctions with single parameter agents" [4], co-authored with Christos Papadimitriou, Kunal Talwar, and Éva Tardos.

Chapter 6: Collusion Since coalitions of agents can sometimes collude to cheat the VCG mechanism, if we are to use this mechanism it would be good to understand which groups can successfully collude, and under which circumstances these opportunities arise. We investigate these issues in the case of a well-studied problem called multicast cost-sharing. This chapter is based on parts of the paper "Approximation and collusion in multicast cost-sharing" [3] co-authored with Joan Feigenbaum, Arvind Krishnamurthy, Rahul Sami and Scott Shenker.

Chapter 2

Preliminaries

This chapter contains background information motivating our model, and tools used in the remaining chapters. Section 2.1 proves that the Vickrey auction is truthful, as we promised in Chapter 1. Section 2.2 discusses why we focus on truthful mechanisms, and includes the famous "revelation principle." Section 2.3 reviews some of the major general results in mechanism design, including the VCG mechanism, mentioned in Chapter 1. Section 2.4 describes the relevant results on constructing truthful mechanisms when the agents have one-dimensional linear valuations, the model we use throughout the remaining chapters. Section 2.5 introduces our notions of truthfulness for randomized mechanisms. Section 2.6 shows how to overcome some difficulties in computing payments. Finally, Section 2.7 discusses some limitations of our model.

Sections 2.1 and 2.4 through 2.6 develop essential tools that are used in the remaining chapters. The other sections exist primarily to provide context, and could be skimmed.

2.1 The Vickrey auction

Recall the Vickrey auction introduced in Section 1.2.1. An auctioneer is selling a single item. Each bidder i has a non-negative valuation t_i for receiving the item, and valuation 0 otherwise. The auctioneer collects the bids b, selects the highest bidder to be the winner, and charges her the amount of the second-highest bid. We had previously left the tie-breaking rule unspecified. This detail is not terribly important, but let us fix a rule for concreteness. We number the agents

from 1 to $|\mathcal{N}|$. If two bidders tie for the highest bid, then the lower-numbered agent is declared the winner. Recall from Section 1.3 that we assume agent i's utility is given by $u_i(b) = t_i q_i(b) - P_i(b)$, where $P_i(b)$ is the amount she pays to the mechanism and $q_i(b) \in \{0, 1\}$ indicates whether i wins.

Theorem 2.1.1 *The Vickrey auction is truthful.*

Proof: We must prove that, for each agent i, setting $b_i = t_i$ is a dominant strategy. To see this, fix i and suppose that the highest bid among all other agents is h. If $h < t_i$, then by bidding t_i, agent i wins the object and pays h, resulting in a positive utility $t_i - h$. In fact, every bid above h yields an identical result: i wins the object at price h, whereas bidding below h will cause i to lose the auction, resulting in a utility of 0. Thus, bidding t_i is an optimal strategy for i in this case. If $h > t_i$, then bidding anything below h will result in a utility of 0, whereas bidding above h will result in a negative utility of $t_i - h$. If $h = t_i$, then agent i is indifferent between losing the auction and winning at price h, so all bids are equally good. Therefore, bidding truthfully in the Vickrey auction is a dominant strategy for agent i, because no matter what the other agents bid, setting $b_i = t_i$ maximizes i's utility. ■

Note that bidding truthfully in the Vickrey auction is the *only* dominant strategy for i, since for every bid $b_i \neq t_i$, there is some value of b_{-i} that will cause this bid to be suboptimal, namely any b_{-i} such that $\max_{j \neq i} b_j$ lies strictly between b_i and t_i.

2.2 Why do we require truthfulness?

A natural question to ask is why we are fixated on truthful mechanisms. This really encompasses two separate issues. First, why do we restrict our attention to mechanisms where the agents' strategies are simply to report their types? Why not instead construct games with more general spaces of strategies, where each agent still has a dominant strategy? Second, why do we desire games with dominant strategies? We address these two issues separately.

First we define precisely what we mean by dominant strategies. Suppose that agent i can select her strategy from an arbitrary set $\mathcal{A}_i$. In the special case that $\mathcal{A}_i = \mathcal{T}_i$, as we have been using so far, then the mechanism is called a *direct revelation* mechanism. Let $\mathcal{A}_{-i} = \prod_{j \neq i} \mathcal{A}_j$, and $\mathcal{A} = \prod_{i \in \mathcal{N}} \mathcal{A}_i$. Then the output and payments are now functions of $a \in \mathcal{A}$, the vector of strategies chosen by the agents. Analogous to our notation for bids, we let a_{-i} denote the vector of strategies chosen by all agents besides i, and write the vector of chosen strategies as either a or (a_{-i}, a_i), as is convenient. A strategy $a_i^* \in \mathcal{A}_i$ is a dominant strategy for agent i if $u_i(a_{-i}, a_i^*) \geq u_i(a_{-i}, a_i)$ for all $a_{-i} \in \mathcal{A}_{-i}$ and $a_i \in \mathcal{A}_i$. As usual, u_i depends implicitly on agent i's type, so we really mean that a_i^* is a dominant strategy for agent i when her type is t_i.

A function $f : \mathcal{T} \to \mathcal{O}$ is called a *social choice function.* It specifies the outcome desired by the mechanism designer, when the type vector is t. The social choice function f is said to be *implementable* in dominant strategies if there exist an output function $O : \mathcal{A} \to \mathcal{O}$ and accompanying payment functions $P_i : \mathcal{A} \to \mathbb{R}$ such that, for all $i \in \mathcal{N}$ and $t_i \in \mathcal{T}_i$, agent i has a dominant strategy $a_i^*(t_i)$ in the mechanism (O, P), and $O((a_i^*(t_i))_{i \in \mathcal{N}}) = f(t)$. In words, when each agent plays her dominant strategy, the mechanism gives the desired output.

A result called the *revelation principle* says that every social choice function implementable in dominant strategies can be implemented by a truthful direct revelation mechanism. The idea behind this is to simulate any mechanism with dominant strategies by using a direct revelation mechanism.

Theorem 2.2.1 (Revelation principle) *If there is a mechanism $\mathcal{M}$ that implements f, then there is also a truthful direct revelation mechanism $\mathcal{M}'$ that implements f.*

Proof: We construct a mechanism $\mathcal{M}'$ to simulate $\mathcal{M}$. Let $a_i^*(t_i)$ denote one dominant strategy for agent i when her type is t_i. In the direct revelation mechanism $\mathcal{M}'$, let b result in the same outcome and payments that the vector of strategies $(a_i^*(b_i))_{i \in \mathcal{N}}$ produced in $\mathcal{M}$. In $\mathcal{M}'$, clearly it is a dominant strategy for agent i to bid $b_i = t_i$, because this is equivalent to playing the dominant strategy $a_i^*(t_i)$ in $\mathcal{M}$. ■

We will be concerned with what social choice functions can and cannot be implemented in dominant strategies, and so we focus only on finding truthful direct revelation mechanisms. However, one should note that in practice it is sometimes useful to use indirect mechanisms. For instance, an agent may have to do some computation in order to figure out her valuation for a good being auctioned. In this case, it may be easier for the agent to respond to a series of questions of the form "Is your valuation higher than h?" than to answer the question, "What is your valuation?" In some cases an agent's type could come from a very complicated space, so describing it explicitly could be burdensome. This will not be an issue for us, since all of the types we deal with are just single real numbers.

We now come to the second question: why do we require dominant strategies? In game theory, there are many theories about how agents will react to a strategic

situation. An assertion about the combination of strategies that a group of agents will play is called an *equilibrium concept* or *solution concept*. Whenever we apply a solution concept, we should ask ourselves how likely it is that agents will actually play according to this prediction. The concept of dominant strategies is the most convincing one.

In the bulk of the game theory literature, dominant strategies do not receive much attention, because in most naturally-occurring games, they do not exist. In fact, some would even argue that games where dominant strategies do exist are not mathematically interesting, because it is obvious what each agent should do. One could study mechanism design using some other solution concept, such as Nash equilibrium, Bayesian Nash equilibrium, or the serially undominated set (see, e.g., [59, Chapter 10]). But the point of mechanism design is to accomplish some task, not necessarily to create interesting games. Whereas most of game theory tries to gain a grasp on how agents will behave in the face of uncertainty about the other agents' choice of strategies, mechanism design attempts to *create* games where the agents' actions will be predictable. Whenever it is possible to design mechanisms with dominant strategies, this is the best choice.

Another argument for designing mechanisms with dominant strategies is that it makes the game much easier for the agents to play. For instance, in an auction setting, each agent only has to compute what the good is worth to her, and need not worry about doing any research into how the other agents are likely to play. This was cited by the U.S. Treasury as one of the primary reasons they adopted a variant of the Vickrey auction [43].

For these reasons, we stick with dominant strategies as our solution concept.

2.3 General constructions and impossibility results

In this section, we survey the main results of mechanism design, concerning what truthful mechanisms we can design, depending on how much we assume about the form of the agents' utilities. To summarize, the Gibbard-Satterthwaite theorem says that there are no interesting truthful mechanisms if the valuations are arbitrary and the mechanism is not allowed to make side payments. If side payments are allowed and the valuations are arbitrary, then the VCG mechanism is truthful, but Roberts's theorem says VCG is essentially the only option. Thus, to create other mechanisms, we must make further assumptions about the form of the valuations. We take this up in Section 2.4.

2.3.1 The Gibbard-Satterthwaite theorem

In this section, we consider trying to create truthful mechanisms without using side payments. In this case, the only useful information that the utility function u_i gives us is the relative order of agent i's preferences. The actual numbers $u_i(o)$ for $o \in \mathcal{O}$ are irrelevant. So we represent u_i with a preference relation $\succeq_i$. The assertion $x \succeq_i y$, where $x, y \in \mathcal{O}$ means "agent i likes outcome x at least as well as outcome y." This is equivalent to $u_i(x) \geq u_i(y)$. Similarly, $x \succ_i y$ is equivalent to $u_i(x) > u_i(y)$. A preference relation $\succeq_i$ is said to be *strict* if no two outcomes are considered equally good. That is, for every pair of outcomes x and y, we have either $x \succ_i y$ or $y \succ_i x$. In this case, we denote the preference relation by $\succ_i$ instead of $\succeq_i$. Let $\mathcal{P}$ denote the set of all strict preference relations over $\mathcal{O}$. A *preference profile* $\succ$ is a vector of preference relations, one for each agent. Suppose that the type space $\mathcal{T}_i = \mathcal{P}$ for each agent i. Then $\mathcal{T} = \mathcal{P}^{\mathcal{N}}$. There is one uninteresting sort of truthful mechanism called a *dictatorship*.

Definition 2.3.1 *An output function $O : \mathcal{P}^{\mathcal{N}} \to \mathcal{O}$ is* dictatorial *if there exists an agent $i \in \mathcal{N}$ such that for every preference profile $\succ$, we have $O(\succ) = o$, where o is the outcome that $\succ_i$ ranks highest.*

In other words, a dictatorial choice rule ignores the preferences of all agents besides i, and just selects i's favorite outcome. This mechanism is truthful, because for all agents $j \neq i$, all strategies are equally bad, and agent i is best off telling the truth, since the mechanism just chooses the result that she says is her favorite. Clearly, such a mechanism is undesirable. The bad news is that we cannot do any better.

Theorem 2.3.2 (Gibbard [26], Satterthwaite [65]) *Suppose that the range of O consists of at least three possible outcomes, and O yields a truthful mechanism. Then O is dictatorial.*

We refer the reader to [45, pp. 873-876] for a proof. This result feels very similar to Arrow's impossibility theorem for voting systems (see [7] or [45, pp. 792-799]), and in fact the standard proof uses Arrow's theorem.

2.3.2 The Vickrey-Clarke-Groves mechanism and Roberts's theorem

The Gibbard-Satterthwaite theorem motivates us to investigate what types of truthful mechanisms are available if we allow side payments. This leads us to the VCG mechanism mentioned in Chapter 1. This and its weighted variant are always truthful. Surprisingly, these are the only truthful mechanisms, if we allow the valuation functions to be arbitrary. Throughout this section, we identify an agent's type with her valuation function.

The Vickrey auction (Section 2.1) gives some clues as to how a truthful mechanism must look. In the Vickrey auction, notice that the price an agent pays is independent of her own bid, up to whether she wins or loses: every bid above some threshold T causes her to win and pay T, whereas every bid below T causes her to lose and pay nothing. Indeed, this principle holds in general.

Proposition 2.3.3 (Bid-independence principle) *If the mechanism (O, P) is truthful and $O(b_{-i}, b_i) = O(b_{-i}, b'_i)$, then $P_i(b_{-i}, b_i) = P_i(b_{-i}, b'_i)$.*

Proof: Assume on the contrary that $P_i(b_{-i}, b_i) < P_i(b_{-i}, b'_i)$. When $t_i = b_i$ and the other agents bid b_{-i}, agent i is better off to lie by bidding b'_i. ■

Thus, $P_i(b)$ can be thought of as $P_i(b_{-i}, O(b_{-i}, b_i))$. That is, i's payment depends on her bid only insofar is it affects the outcome chosen.

The biggest positive result in mechanism design is the Vickrey-Clarke-Groves (VCG) mechanism [69, 14, 31] (see also [45, Chapter 23]), a generalization of the Vickrey auction. For this mechanism, each agent i's type consists of her valuation function v_i, so b_i is the valuation function that i reports. The VCG mechanism can be used when the goal of our mechanism is to maximize the total valuation of the agents.

The VCG mechanism (O, P) is given by:

$$\begin{aligned} O(b) &= o^* \text{ where } o^* \in \operatorname{argmax}_{o \in \mathcal{O}} \sum_{i \in \mathcal{N}} b_i(o) \\ P_i(b) &= \sum_{j \neq i} b_j(o) + h_i(b_{-i}) \end{aligned}$$

where the functions h_i are arbitrary. That is, VCG selects the outcome that maximizes the total reported valuation. Notice that, as required by the bid-independence principle, the payment P_i depends on b_i only through its influence

on the outcome o^*. The h_i terms in the payments are irrelevant to truthfulness, because nothing that agent i does can change $h_i(b_{-i})$. The $\sum_{j \neq i} b_j(o)$ term is the ingenious bit, because it serves to make the utility of a truthful bidder exactly equal to the utilitarian objective function (modulo the h_i term).

Theorem 2.3.4 *The VCG mechanism is truthful.*

Proof: Agent i's utility in the VCG mechanism is $(\sum_{j \neq i} b_j(o^*) + v_i(o^*)) + h_i(b_{-i})$. Since i's bid b_i has no effect on the h_i term, she should select her bid so as to maximize the first term, via the choice of o^* that it induces. Suppose that o is the outcome that maximizes $\sum_{j \neq i} b_j(o) + v_i(o)$. Agent i can cause the mechanism to choose this outcome (or an equally good one) by bidding $b_i = v_i$, because then the function that the mechanism maximizes is exactly this term. If agent i submits a different bid b_i, then the mechanism will select an outcome o^* that maximizes $\sum_{j \neq i} b_j(o^*) + b_i(o^*)$, but does not necessarily maximize $\sum_{j \neq i} b_j(o^*) + v_i(o^*)$. ■

Recall that the objective function that VCG optimizes, the sum of the total valuations, is called the utilitarian objective. The VCG mechanism works precisely because it chooses the payment function to line up each agent's utility with the utilitarian objective. One particular choice of the functions h_i results in a particularly nice interpretation. For a subset of agents $S \subseteq \mathcal{N}$, let $V(S) = \max_{o \in \mathcal{O}} \sum_{j \in S} b_j(o)$. If we set $h_i(b_{-i}) = -V(\mathcal{N} - i)$, then agent i's overall utility in the VCG mechanism when she bids truthfully is $u_i = V(\mathcal{N}) - V(\mathcal{N} - i)$. This can be interpreted as the net benefit to society that results from the presence of agent i. In most applications, $V(S)$ is increasing in the set S, and so a truth-telling agent never incurs negative utility.

The basic VCG mechanism can be modified by weighting the agents differently and adding a bias to the outcome function, while preserving truthfulness [30, 64].

Specifically, let $w \in \mathbb{R}_+^{\mathcal{N}}$ be a set of non-negative weights. Let $H : \mathcal{O} \to \mathbb{R}$ be a "bias" function. The resulting *weighted, biased VCG mechanism* is defined by

$$O(b) = o^* \qquad \text{where } o^* \in \text{argmax}_{o \in \mathcal{O}}(\sum_{i \in \mathcal{N}} w_i b_i(o) + H(o))$$

$$P_i(b) = \frac{1}{w_i}\left(\sum_{j \neq i} w_j b_j(o*)\right) + h_i(b_{-i}) \quad \text{when } w_i > 0$$

$$P_i(b) = h_i(b_{-i}) \qquad \text{when } w_i = 0,$$

where the choice of o^* does not depend on any b_i such that $w_i = 0$. This last condition is just a technical nuisance to keep an agent i with $w_i = 0$ from benefitting by lying.

Theorem 2.3.5 *For every choice of weights and bias function, the weighted, biased VCG mechanism is truthful.*

The proof is essentially the same as the proof of Theorem 2.3.4. Surprisingly, these are the only truthful mechanisms, if we allow valuations to be arbitrary.

Theorem 2.3.6 (Roberts [64]) *Suppose each agent i's valuation is allowed to be any function $v_i : \mathcal{O} \to \mathbb{R}$, the mechanism (O, P) is truthful, the range of O is all of $\mathcal{O}$, and $|\mathcal{O}| \geq 3$. Then the mechanism is a weighted, biased VCG mechanism.*

By the range of O, we mean the set $\{O(t) \in \mathcal{O} : t \in \mathcal{T}\}$. Notice that if the range of O is less than the entire co-domain $\mathcal{O}$, then we can restrict the co-domain to be range(O), and then apply Theorem 2.3.6.

The VCG mechanism is good because it handles general valuations. However, if the mechanism designer wishes to optimize some function other than slight variants on the utilitarian objective, then it is useless. Because of Theorem 2.3.6, if we want to optimize any other functions, we will have to restrict the form of the agents' valuations.

2.4 One-dimensional linear valuations

As we mentioned in Section 1.3, our work will assume that the agents have one-dimensional linear valuations. Recall that this means the outcome of the mechanism will be a vector of allocations $(q_i)_{i \in \mathcal{N}}$, agent i's type t_i is a non-negative real number, and her valuation is given by $v_i = \pm t_i q_i$. Thus, her type represents her cost or benefit per unit of allocation. The quantity q_i might represent some amount of work that the agent must perform, or an amount of a good that she gets to consume.

Roughly, the main theorem is that an output function is implementable if and only if the allocation to agent i is monotone in i's bid. In the case that $v_i = t_i q_i$, the allocation should be increasing; in the case that $v_i = -t_i q_i$, the allocation whould be decreasing. We show a standard proof of this fact in Section 2.4.2, along with an alternate simpler pictorial proof of ours in Section 2.4.3.

One-dimensional linear valuations are actually a special case of a somewhat more general result proved by Mirrlees [49] in 1971 and extended in [55, 10, 46, 32]. This result concerns agents whose utility functions satisfy a condition called the *single-crossing property*, and says roughly that an output function is implementable if and only if the allocations are monotone. We refer the reader to [25, pp. 257-262] for a full discussion.

We first examine the case of binary allocations ($q_i \in \{0, 1\}$), then move on to the case of real allocations ($q_i \geq 0$).

2.4.1 Binary allocations

Suppose that the quantities q_i are binary, and $v_i(q_i) = -t_i q_i$. In this case, for $q_i(b_{-i}, b_i)$ to be a decreasing function of b_i implies that there is a single threshold

value where q_i jumps from 1 to 0. We will show that the mechanism is truthful if and only if it has such thresholds.

For each b_{-i}, suppose there is some *threshold bid* $T_i(b_{-i})$ such that if $b_i < T_i(b_{-i})$ then i "wins" (i.e., $q_i = 1$), while if $b_i > T_i(b_{-i})$ then i loses (i.e., $q_i = 0$). If $b_i = T_i(b_{-i})$, then we do not care whether i wins or loses. If i loses, she is paid $P_i(b_{-i}, 0)$. If she wins, she is paid $P_i(b_{-i}, 0) + T_i(b_{-i})$. We call such a mechanism a *threshold mechanism.*

Theorem 2.4.1 *A mechanism with binary allocations is truthful if and only if it is a threshold mechanism.*

Proof: The proof that every threshold mechanism is truthful is essentially identical to the proof of Theorem 2.1.1, that the Vickrey auction is truthful.

Conversely, suppose that the mechanism is truthful. Assume for a contradiction that there exist $b_{-i} \in \mathcal{T}_{-i}$ and bids $T_1 < T_2$ for i such that $q_i(b_{-i}, T_1) = 0$ and $q_i(b_{-i}, T_2) = 1$. Let $\Delta P = P_i(b_{-i}, T_2) - P_i(b_{-i}, T_1)$. When i's true type is T_1, then it cannot be to her advantage to report T_2, so $\Delta P \leq T_1$. That is, the extra payment she would receive is no more than the extra cost she incurs. When i's true type is T_2, then it cannot be to her advantage to report T_1, so $\Delta P \geq T_2$. This is a contradiction. Thus, the mechanism has threshold bids.

By the bid-independence principle (Proposition 2.3.3), $P_i(b)$ depends only on b_{-i} and $q_i(b)$. Let us fix b_{-i}, and let $\Delta P = P_i(b_{-i}, T_1) - P_i(b_{-i}, T_2)$, where $T_1 < T_i(b_{-i}) < T_2$. When i's true type is T_1, it cannot be to her advantage to report T_2, so $\Delta P \geq T_1$. When i's true type is T_2, it cannot be to her advantage to report T_1, so $\Delta P \leq T_2$. This holds for all $T_1 < T_i(b_{-i}) < T_2$, so $\Delta P = T_i(b_{-i})$. ■

Definition 2.4.2 *A mechanism satisfies the voluntary participation condition if an agent who reports her type truthfully is always guaranteed a non-negative utility.*

A threshold mechanism achieves the voluntary participation property if and only if $P_i(b_{-i}, 0) \geq 0$. Since the mechanism typically wants to minimize its payments, we might as well take $P_i(b_{-i}, 0) = 0$. Thus, the payment to a winning agent i exactly equals her threshold bid $T_i(b_{-i})$.

Thus, for the case of binary allocations, the allocation function is implementable if and only if $q_i(b_{-i}, b_i)$ is monotone decreasing in b_i, for all b_{-i}. Analogously, in the case where $v_i = +t_i q_i$, the allocation function is implementable if and only if $q_i(b_{-i}, b_i)$ is monotone increasing in b_i, for all b_{-i}.

2.4.2 Real allocations

The case of real allocations $q_i \geq 0$ is more complicated, but yields the same monotonicity result in the end. If $v_i(q_i) = -t_i q_i$, then the allocation is implementable if and only if $q_i(b_{-i}, b_i)$ is monotone decreasing in b_i, for all b_{-i}. And if $v_i(q_i) = +t_i q_i$, then the allocation is implementable if and only if $q_i(b_{-i}, b_i)$ is monotone increasing in b_i, for all b_{-i}.

Since we are arguing about dominant strategies, the characterization problem actually reduces to the single-agent case. For fixed b_{-i}, $q_i(b_{-i}, b_i)$ becomes just a function of the bid b_i. Thus, for the rest of this section we shall argue about a single agent with a type $t \in \mathbb{R}$, who bids a number $b \in \mathbb{R}$, causing herself to receive allocation $q(b)$ and payment $P(b)$, resulting in an overall utility of $u = P(b) - tq(b)$. For the symmetric case where the utility is $u = +tq(b) - P(b)$, the derivation is similar.

Derivation of monotonicity result for arbitrary loads. Here we present a standard argument, following the presentation in [39, pp. 63-65]

Suppose we have a truthful mechanism (q, P) for a single agent. We start by

deriving necessary conditions for truthfulness, which turn out to also be sufficient. Let us define $U(t) = -tq(t) + P(t)$. Thus, $U(t)$ denotes the agent's utility when she bids her true type. Truthfulness is equivalent to, for all $b, t \geq 0$,

$$\begin{aligned} U(t) &\geq P(b) - tq(b) \\ U(t) &\geq P(b) - bq(b) + bq(b) - tq(b) \\ U(t) &\geq U(b) + (t-b)(-q(b)). \end{aligned} \tag{2.1}$$

Thus, $U(t)$ is a convex function, and $-q(t)$ is a subgradient at t. Standard results from analysis show that $U(t)$ is continuous on $[0, \infty)$, differentiable almost everywhere, and is equal to the integral of its derivative. Also, $U'(t) = -q(t)$ wherever U is differentiable. Thus, $U(t) = U(0) - \int_0^t q(x)dx$. Writing $U(t) = -tq(t) + P(t)$ and rearranging gives

$$P(t) = P(0) + tq(t) - \int_0^t q(x)dx. \tag{2.2}$$

Since $U(t)$ is convex, its derivative is increasing, which implies that $q(t)$ is decreasing.

Thus, for the mechanism to be truthful, it is necessary that $q(t)$ is decreasing and $P(t)$ is given by equation (2.2). We can easily verify that these conditions are also sufficient. To check this, note that the truthfulness conditions are equivalent to inequality (2.1) holding for all $b, t \geq 0$. Using equation (2.2) and the definition of U, this condition rearranges to $\int_b^t q(x)dx \leq (t-b)q(b)$. This holds because $q(b)$ is decreasing. Thus, we have proven the following result.

Theorem 2.4.3 *Consider a mechanism (q, P) for a single agent of type t, where $u(b) = -tq(b) + P(b)$. The mechanism is truthful if and only if $q(b)$ is decreasing in b, and*

$$P(b) = P(0) + bq(b) - \int_0^b q(x)dx \tag{2.3}$$

In the event that $\int_0^\infty q(x)dx < \infty$, we can set $P(0) = \int_0^\infty q(x)dx$, and the payment formula reduces to $P(t) = tq(t) + \int_t^\infty q(x)dx$. In this case $U(t) = \int_t^\infty q(x)dx > 0$, so we satisfy the voluntary participation property as well. As mentioned previously, this extends easily to the case of multiple agents.

Theorem 2.4.4 *Consider a mechanism (q, P) for a single agent of type t, where $u(b) = -tq(b) + P(b)$. Given a monotone decreasing allocation function $q(b)$, we can choose payments to implement q and satisfy the voluntary participation condition if and only if*

$$\int_0^\infty q(x)dx < \infty. \tag{2.4}$$

In the case that this integral is finite, then setting

$$P(t) = tq(t) + \int_t^\infty q(x)dx \tag{2.5}$$

is the minimum payment function that implements q and satisfies the voluntary participation condition.

Proof: We have already noted that setting the payment as indicated guarantees truthfulness and the voluntary participation condition, in the case that (2.4) holds. Now suppose that the payment function P implements q and satisfies voluntary participation. By Theorem 2.4.3, the payment function must satisfy $P(b) = P(0) + tq(t) - \int_0^t q(x)dx$. Thus, $U(t) = P(0) - \int_0^t q(x)dx$. If we are to satisfy the voluntary participation condition, then we must have $\int_0^t q(x)dx \leq P(0)$ for all t. Thus, $\int_0^\infty q(x)dx < \infty$, and the minimum value we can choose for $P(0)$ is $\int_0^\infty q(x)dx$. ■

An analogous theorem with an analogous proof holds for the case where $v = +tq$. We state this theorem here for future reference. Notice that in this case the mechanism is collecting a payment from the agent, so we would typically want to maximize the payment.

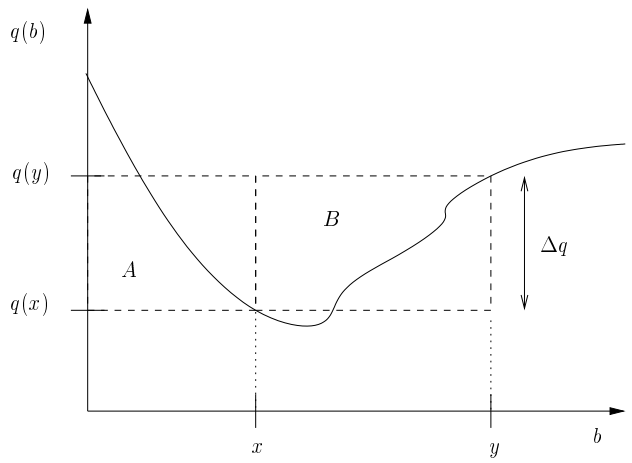

Figure 2.1: This graph shows why we cannot allow the load to increase with b.

Theorem 2.4.5 *Consider a mechanism (q, P) for a single agent of type t, where $u(b) = +tq(b) - P(b)$. The allocation function $q(b)$ is implementable if and only if it is monotone increasing. Given such an allocation function, the maximum payment function that satisfies the voluntary participation condition and yields a truthful mechanism is*

$$P(b) = P(0) + tq(t) - \int_0^t q(x)dx. \tag{2.6}$$

2.4.3 Pictorial proofs

We now give an alternate simple pictorial proof of Theorem 2.4.3.

Proof of Theorem 2.4.3: We explain the pictorial proofs of Figures 2.1 and 2.2. In Figure 2.1, A and B denote the areas of the rectangles they label. If i's true value is y, she would save cost $A + B$ by bidding x. If her true value is x, she would incur an extra cost of A by bidding y. To motivate truth-telling, the extra payment for bidding y instead of x should be at least $A + B$ and at most A, which is impossible since $B > 0$. Therefore, the allocation curve must decrease monotonically.

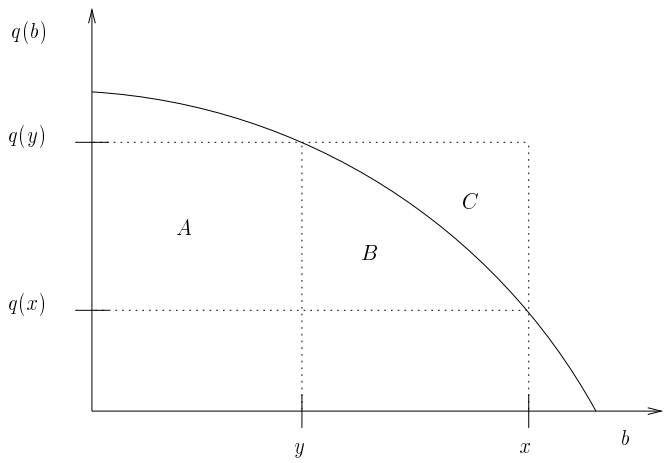

Figure 2.2: This picture shows why agent i never gains by overbidding.

In Figure 2.2, the allocation curve is decreasing and the payments are given by (2.2). Geometrically, the payment to i if she bids x is a constant minus the area between the allocation curve and the horizontal line at height $q_i(x)$. If agent i's true value is y and she bids $x > y$, then her cost decreases by A from the decreased allocation, but her payment decreases by $A + B$. Since $B \geq 0$, she never benefits from overbidding. Similarly, she never benefits from underbidding. Thus, the given payment function does induce truth-telling.

It remains to be shown that the payments must be of the given form. From Figure 2.2, we see that the payment must decrease by at least A and at most $A + B + C$ when the bid changes from y to x. The lower bound arises in order to keep an agent of type y from wanting to overbid, and the upper bound arises in order to keep an agent of type x from wanting to underbid. Subdividing the interval $[y, x]$ and making a limiting argument show that the payment formula must be of the form given by (2.3). ■

2.5 Randomized mechanisms

One very useful technique in algorithmic design is the use of randomness. A randomized algorithm combines an ordinary input with the outcomes of some random coin flips in order to determine its output. See [51] for a good introduction. In the context of mechanism design, we must extend our notion of dominant strategy truthfulness to accommodate the randomness. We discuss three ways to do this.

Universal truthfulness. One way to view a randomized mechanism $\mathcal{M}$ is that it selects a random ω from a probability space Ω, and then uses a deterministic mechanism $\mathcal{M}_\omega$. The randomized mechanism is said to be *universally truthful* if every one of the mechanisms $\mathcal{M}_\omega$ in the support of the randomized mechanism is truthful. In other words, even if agent i knows both the other bids b_{-i} and the coin flips ahead of time, she cannot benefit by lying. This concept was proposed in [58] and has also been used successfully in, e.g., [9, 23, 28]. However, sometimes this notion is too strong, to be obtainable so we consider two ways to weaken it slightly.

Truthfulness in expectation. Suppose that for each agent i and each possible set of bids $b_{-i} \in \mathcal{T}_{-i}$ for the other agents, $E[u_i(b_{-i}, t_i)] \geq E[u_i(b_{-i}, b_i)]$ for all $b_i, t_i \in \mathcal{T}_i$. That is, bidding truthfully always maximizes agent i's expected utility, no matter what the other agents do. Then we say that the mechanism is *truthful in expectation*. If agent i knows the other agents' bids ahead of time but not the coin flips, then she cannot benefit from lying, if her goal is to maximize her expected utility. Theorems 2.4.3, 2.4.4 and 2.4.5 still hold, but with q replaced by the expected allocation, and P replaced by the expected payment.

Truthfulness with high probability. Suppose that an agent might sometimes benefit by lying, but only with small probability. Formally, suppose that for each b_{-i} and t_i, $Pr[t_i \notin \text{argmax}_{b_i} u_i(b_{-i}, b_i)] \leq \epsilon$. Then we say the mechanism is *truthful with error probability* ϵ. If ϵ is inverse polynomial in some specified parameters of the problem (such as the number of agents), then we say that the mechanism is *truthful with high probability*. Even in the rare event that a bad ω is chosen, computing an effective lie could be difficult for agent i and would typically require knowledge about the other bids. Moreover, such a lie might backfire in the probability $(1 - \epsilon)$ event that the mechanism selects a good ω. In using such a mechanism, one hopes that these factors combined will convince the agents not to bother lying. This notion may be preferable to that of truthfulness in expectation because it does not assume anything about how the agents view probability distributions over outcomes.

Two comments are in order here. In much of game theory, one uses von Neumann-Morgenstern (VNM) utility functions. These define an agent's utility not only for deterministic outcomes, but also for probability distributions over outcomes (often called *lotteries*). VNM utilities, by definition, satisfy the property that an agent's utility for a randomization over deterministic outcomes equals her expected utility. (See [45, Chapter 6] for an in-depth discussion of the theory of utilities.) In our work we do not assume that agents place utilities on lotteries. Instead, we use the concepts of truthfulness in expectation and truthfulness with high probability. If an agent does have a utility function over lotteries (meaning that she will always act so as to attempt to obtain the lottery that maximizes this function) and that utility function happens to be a VNM utility, then bidding truthfully is a dominant strategy in any mechanism that is truthful in expectation.

In some cases (such as the mechanism developed in Section 4.2) we are able to design mechanisms where truthful bidding guarantees a fixed utility with probability 1, whereas other bids both have lower expected utility and non-zero variance. In some models of portfolio management, the agent's objective is to maximize $\mu - r\sigma$, where μ is the expected profit, σ^2 is the variance, and r is a (usually non-negative) parameter describing the agent's attitude about risk. The case $r = 0$ denotes a risk-neutral agent, which is the same as assuming that profit serves as a VNM utility function. Our zero-variance mechanism is truthful for agents with any non-negative value of r. Thus, in these mechanisms, truth-telling is a dominant strategy under a strictly weaker assumption than that of VNM utilities.

Our second comment is that randomness introduced by the mechanism is fundamentally different than randomness that appears in a Bayesian framework. This is because the distributions used to generate the random variables used is made public knowledge (along with the output and payment functions) before the agents have to make their bids. Agents do not have to formulate probabilistic models about the state of nature, or about the other agents' beliefs.[1] Nor do they need to have full knowledge of the game. They just need to know that truthful bidding is a dominant strategy.

2.6 Tricks for computing payments

In this section we discuss how to compute the payment formulas from Theorems 2.4.4 and 2.4.5, in cases where it is not straightforward. Figures 2.3 and 2.4 illustrate the payment formula, in the two cases where $v = -tq$ and $v = +tq$. Both

[1]They do have to trust that the mechanism really is generating its random variables according to its published rules. This raises an interesting practical problem of how one might verify this.

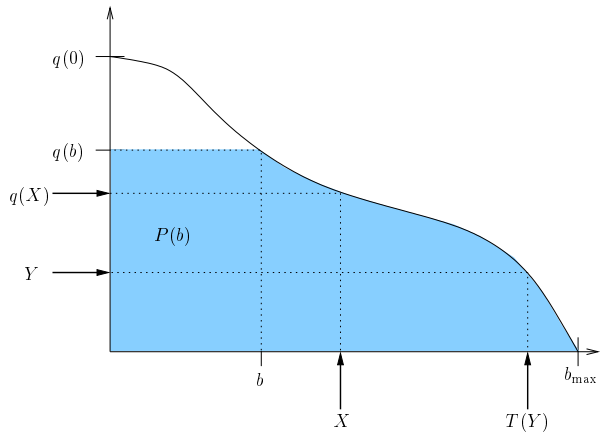

Figure 2.3: The shaded area gives the payment, in the case where $v(q) = -tq$.

involve definite integrals of the function $q(b)$. If we are lucky, then we can derive an explicit form for $q(b)$ as a function of b, and integrate it symbolically. Otherwise, we have to employ some tricks in order to compute the payments in polynomial time.

Sometimes we know the shape of the curve, but it has exponentially many breakpoints. Sometimes we may not even know the shape of the curve. Another issue is what to do when $\int_0^\infty q(x)dx = \infty$. In the first case, we can either discretize the bids, or use random sampling. Random sampling can also help in the second case. In the third case it can help to create an artificial cutoff bid. We describe these tricks below. We discuss them for the case $v = -tq$ (Figure 2.3). Analogous tricks hold for the case $v = +tq$ (Figure 2.4).

Discretizing the bids. We can select a constant $\epsilon > 0$ and round up every bid to the next power of $(1 + \epsilon)$. In most optimization settings, this will only throw off our objective function by a factor of at most $(1 + \epsilon)$. This does not destroy truthfulness, because the allocation curve is still monotone – we have changed it into a step function that is constant on each segment $((1 + \epsilon)^k, (1 + \epsilon)^{k+1}]$, $k \in \mathbb{Z}$.

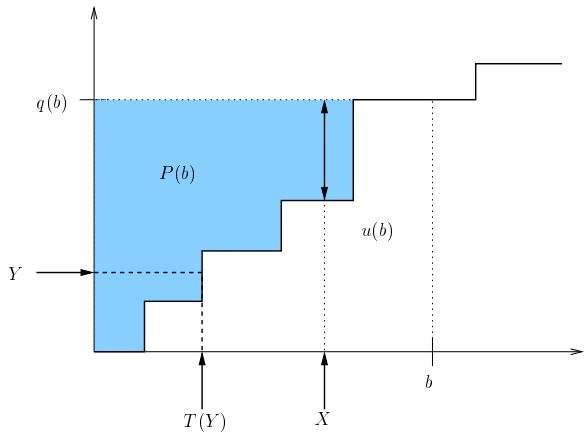

Figure 2.4: The shaded area gives the payment, in the case where $v(q) = tq$.

The advantage is that this step function has at most $\lceil \log_{1+\epsilon} \frac{b_{\text{hi}}}{b_{\text{lo}}} \rceil$ breakpoints, where b_{lo} and b_{hi} are the limits of integration.

Random sampling. If it is acceptable to have a randomized mechanism that is truthful in expectation, we can easily create a randomized payment with the correct expectation.

Suppose, as in Figure 2.3, there is some bid $b_{\max}$ such that $q(b_{\max}) = 0$. Generate a random variable X uniformly in $[0, b_{\max}]$. If $X \in [0, b]$, set the payment to $b_{\max} q(b)$. Otherwise, set the payment to be $b_{\max} q(X)$. The expected payment is then

$$\begin{aligned} E[P(b)] &= \int_0^b b_{\max} q(b) \frac{1}{b_{\max}} dx + \int_b^{b_{\max}} b_{\max} q(x) \frac{1}{b_{\max}} dx \\ &= bq(b) + \int_b^{\infty} q(x) dx \end{aligned}$$

as desired. We call this "sampling from the bid axis." This approach can be modified to work even if the allocation itself is random. In this case, we replace $q(b)$ or $q(X)$ with an unbiased estimator, which we can obtain by running the allocation algorithm once.

An alternate approach is to "sample from the quantity axis." Suppose that, for any Y, we can efficiently compute $\sup\{x : q(x) \geq Y\}$, which we define to be $T(Y)$. Then we can generate a random variable Y uniformly in $[0, q(b)]$ and pay $q(b)T(Y)$. This works even if q never drops to zero, provided that $\int_b^\infty q(x)dx < \infty$. The expected payment is then

$$\begin{aligned} E[P(b)] &= \int_0^{q(b)} q(b)T(y)\frac{1}{q(b)}dy \\ &= \int_0^{q(b)} \int_0^{T(y)} 1 \; dxdy \\ &= bq(b) + \int_b^\infty q(x)dx, \end{aligned}$$

where the last step is obtained by switching the order of integration and simplifying.

Creating an artificial cutoff. If the problem is that $\int_b^\infty q(x)dx = \infty$, then there is no payment rule that will satisfy the voluntary participation condition. However, in some situations we can simply ignore the really high bids above some artificial cutoff without hurting our objective function too much. In this modified mechanism, q drops to zero at this artificial cutoff, so the integral is finite.

Randomized mechanisms with zero-variance utilities. When applying Theorem 2.4.4 to a mechanism with a randomized allocation rule, the curve q will represent an expected load. To achieve truthfulness in expectation, it is enough to create any randomized payment rule that yields the payment (2.5) in expectation. Suppose that we actually can compute this integral deterministically. In this case, we can adjust the payments so that the utility for a truth-telling agent has zero variance, despite the fact that the allocation algorithm is random. To do this, simply compute $P(b)$ according to (2.5). Then check to see how much the agent's actual (random) load differs from the expected load. If the agent got z

units more load than she was supposed to get in expectation, then compensate her with an extra zb units of money. This does not change her expected payment, since $E[z] = 0$. However, if the agent reports her true type t, then this modification exactly cancels out the variation in her valuation. Thus, her utility will always be $\int_t^\infty q(x)dx$, no matter how the random coin flips of the algorithm come out.

2.7 Limitations of our model

In this section, we point out various limitations of our model.

Notice that our definition of valuations rules out the possibility that one agent's valuation can depend on another agent's type. The reason for this is that we want to design dominant strategy mechanisms, which is inherently impossible to achieve if agent i's valuation depends on another agent's type. In an auction for some good whose value is hard to measure, such as an oil field, the type t_i may reflect some beliefs that i has about the value, such as the amount of oil under the ground. In this case, one might want to introduce such a dependence. Our framework fails to model such situations.

This framework also does not take into account the possibility for repeated play. For instance, bidders in an auction for U.S. Treasury bills know that if they lose this week's auction, they can try again next week. Also, in many auction settings, a bidder would care about the resale value of the object she is buying. This is another reason why her valuation might depend on other agents' types, as these might give some indication about the resale value. We explicitly do not model this: we view all of our mechanisms as one-shot games.

Notice that the general model set forth in Section 1.3 allows agent i's valuation to depend on *all* of the allocations q_j, $j \in \mathcal{N}$, but in our work, it depends only on

q_i. For an example where it clearly should depend on all the allocations, consider a television network bidding for rights to broadcast the Super Bowl. Clearly, the company much prefers exclusive broadcast rights to an allocation where both they and other companies are allowed to broadcast.

Chapter 3

Frugality

In this chapter, we are interested in mechanisms that purchase some service from the agents and whose payments are small, by some measure. We describe these qualitatively as being *frugal*. Often, mechanism design views the payments only as an inducement to the agents to bid truthfully, while the mechanism really cares about optimizing some unrelated objective function, such as finding the shortest path in a graph, or balancing load on machines. Of course, the literature on auctions does care about the payments, because the main point of selling something at auction is to make money (for a good text on auction theory, see [39]). Conversely, when the mechanism is paying to perform a task, it makes sense to try to keep the total payment low.

We consider the setting where the mechanism must accomplish some task, and it will hire a team of agents to do it. Each agent performs a fixed service and incurs a fixed cost for performing that service. There exist certain teams of agents who can combine to perform the task, and no team is a subset of another. One special case is where the agents are edges in a network, we wish to send something through the network between nodes s_1 and s_2, and the teams are $s_1 - s_2$ paths. The catch is, an agent's cost is a secret known only to that agent. One can apply the VCG mechanism (see Section 2.3.2) to select the team that can perform the task while incurring the minimum cost *to itself*. However, the total amount that the *mechanism must pay* to the members of this team can be very high. In the network example, the cheapest team is just the shortest path with respect to the edge costs. Nisan and Ronen [56] indeed applied the VCG mechanism to solve this shortest path problem.

One drawback to the VCG mechanism when applied to the shortest path problem is that it can be forced to pay a huge amount, compared not only to the actual cost of the path, but also compared to the cost of the cheapest path that is completely disjoint from it. This raises the question of whether some other truthful mechanism avoids this problem. Roughly, we show that all reasonable path selection mechanisms can be forced to overpay just as badly as VCG in the worst case. Thus, our results are negative. While we phrase everything in terms of paths because of their familiarity, all of our results have analogs for the more general problem of hiring a team.

3.1 The team and path selection problems

In this section, we define the general team selection problem and associated terminology, and specialize it to the path selection problem. We also note the implications of our general results on truthful mechanisms from Section 2.4.1 when applied in this setting.

The general team selection mechanism design problem is defined as follows. As usual, $\mathcal{N}$ denotes the set of agents. Each agent is capable of performing a fixed service. Agent i's type t_i is the cost she incurs for performing that service. Let $\mathcal{F} \subseteq 2^{\mathcal{N}}$ be a collection of the *feasible* subsets of agents. If $S \in \mathcal{F}$, this means that the team of agents S is capable of performing the desired task. The mechanism must select some feasible set of agents. We require our mechanisms to satisfy the voluntary participation property, as we must in order for the question of minimizing payments to make sense. We refer to the team S selected by the mechanism as the *winning* team, and say that agents $i \in S$ on this team *win*, while agents not on the team *lose*.

Recall the results of Section 2.4.1, which show that in every truthful mechanism, for any fixed bids b_{-i}, there exists a threshold bid $T_i(b_{-i}) \geq 0$ for agent i such that if i bids below $T_i(b_{-i})$ it will win, and if i bids above $T_i(b_{-i})$ it will lose. Thus, we assume that there is no agent i who belongs to every feasible team, or else there would be no threshold bid for i. Also, since we are looking for truthful mechanisms that minimize payments subject to satisfying voluntary participation, we can assume that a winning agent receives payment $T_i(b_{-i})$, whereas a losing agent receives zero payment. Therefore, there is no flexibility in our payment scheme. *The payment scheme is entirely determined by the rule we use for selecting the winning team.*

For the path selection problem, the agents are edges in the graph, there is a source s_1 and a sink s_2, and the feasible sets are all possible paths from s_1 to s_2. By convention, edges in a graph are traditionally denoted by the letter e, so we will tend to refer to the agents as e rather than i, as we have in previous chapters. Also, b_P will denote the vector of bids along a path (or any other set of edges) P, and b_{-P} the vector of all bids besides b_P. If e is a cut edge separating s_1 and s_2, then it is on every feasible path. Thus, we assume that there is no cut edge, and hence there exist two edge-disjoint paths from s_1 to s_2.

By the comments above, our path selection mechanisms are entirely determined by our rule for selecting the winning path. To enforce truthfulness, we need a path selection rule that is *monotone*, meaning that a losing edge can never cause itself to win by raising its bid, i.e., it induces threshold bids for the agents. The payment to a winning edge e will then be the largest amount it could have bid and still won. Recall from Section 2.3.2 that the VCG mechanism selects the shortest path. Thus, the payment to a winning edge e in the VCG mechanism is the maximum amount

that e could have bid and still been on the shortest path. Another interpretation of this is that e's payment exceeds its bid by a bonus equal to its "marginal value to society," the amount by which the shortest path value would increase if e were removed. Notice that in the case that the graph consists of just some parallel arcs from s_1 to s_2, the VCG mechanism for shortest paths reduces to the procurement version of the Vickrey auction, from Section 1.2.1: the lowest bidder wins, and is paid the amount of the second-lowest bid.

Note that the edges are allowed to bid any positive real number, but not zero or ∞. However, we will come across cases where a bid is so large that it is "effectively" infinite, and we will use "bidding ∞" as a convenient shorthand for this. We will make this concept precise in Section 3.4.

3.2 How should we measure frugality?

The procurement version of the Vickrey auction has a nice property that we would hope to carry over to the shortest path problem. If we are buying a good or service from the lowest bidder, we would like to know that we are not losing too much by paying the second lowest bid. Of course, if the lowest bid is 1 and the second lowest is 100, then we are paying a big premium to enforce truthfulness. Since there are other truthful auction mechanisms, we might hope to find one that never pays more than some constant times the lowest bid. Unfortunately, elementary arguments show that this is impossible. However, since Vickrey pays the second lowest bid, if there is tight competition in the market then the premium for truthfulness is probably not high, because the two lowest bids are likely to be close together.

In contrast, the VCG shortest path mechanism can pay many times the cost of

the chosen path, even if there is a disjoint alternate path of similar cost. Consider a graph consisting of two disjoint $s_1 - s_2$ paths, P and Q. Suppose the length of path P (according to the bids) is L, the length of Q is $L(1+\epsilon)$, and P has k edges. Then the VCG mechanism chooses the path P. For each edge $e \in P$, the payment is $b_e + \epsilon L$, since this is the highest it can bid before Q becomes the shorter path. Thus, the total payment made to P is $L(1 + k\epsilon)$. If $k\epsilon$ is large, then we could pay a large factor times the actual cost of the chosen path. Moreover, for fixed ϵ, we pay $\Theta(k)$ times the cost of the *longer* path. The problem here is that the removal of any edge of P forces us to take path Q. So, given the presence of the other edges, the marginal "added value" of an edge $e \in P$ is ϵL, even though it actually requires all of these edges to reap this benefit. If this entire path were one entity, then the VCG mechanism would pay it a profit of only ϵL, but since it is composed of k entities each of whom must be induced to bid truthfully, the mechanism must pay a total profit of $k\epsilon L$.

3.3 Our results

We show that no "reasonable" truthful mechanism avoids this pitfall of the VCG mechanism. Since the payment scheme is entirely determined by the path selection rule, if we restrict ourselves to selecting the shortest path, then VCG is the *only* truthful mechanism. However, it is conceivable that choosing an almost-shortest path might yield a better payment. Since each edge on the chosen path is paid at least its bid, we will still want to choose a fairly short path (with respect to the bids). The VCG mechanism paid a lot because it had to pay profits to many different edges, which suggests that we should give preference to paths with fewer edges.

A natural class of selection rules that allows us to easily encode such a bias is what we call *min function* mechanisms. For each path, we evaluate some function of the bids on its edges, and select the path with the minimum function value. The VCG mechanism is a special case, where the function is just the sum of the bids. Unfortunately, an example similar to the bad example for VCG shows that min function mechanisms can be made to perform similarly poorly.

Since min function mechanisms exhibit essentially the same bad behavior as VCG, it is logical to look for other mechanisms. We outline some basic natural properties that we argue any reasonable path selection rule should have, and verify that min function mechanisms satisfy them. We then prove a characterization result that on two reasonably large classes of graphs, every mechanism satisfying our reasonable properties is indeed a min function mechanism. One corollary of our result is that on any graph containing three node-disjoint $s_1 - s_2$ paths, every mechanism satisfying our properties can be forced to exhibit the same bad behavior as the VCG mechanism. The two classes of graphs for which we prove our result are: (a) graphs with an $s_1 - s_2$ arc, and (b) connected graphs containing s_1 and s_2, with the addition of two node-disjoint $s_1 - s_2$ paths. We also show that our characterization result fails on any graph consisting of only two node-disjoint $s_1 - s_2$ paths. We conjecture that it holds for all graphs containing three node-disjoint $s_1 - s_2$ paths.

Bikhchandani et al. [11] study the behavior of the VCG mechanism applied to the minimum spanning tree (MST) problem. This is the team selection problem where the collection of feasible sets $\mathcal{F}$ consists of spanning trees of the graph. They prove that if T_1 is the MST and T_2 is the MST in the remaining graph after T_1 has been removed, then the VCG mechanism does not pay more than $\text{cost}(T_2)$. This

result contrasts starkly with our main result that, under suitable assumptions, truthful path selection mechanisms can be forced to pay $\Theta(n)$ times the actual costs incurred, even in a tight market.

3.4 Desired properties for truthful mechanisms

We discuss some properties that we argue any reasonable path selection rule ought to possess. First, each edge and path should be able to control its own fate in a limited way. In an ordinary auction (with the agents selling), it is a standard requirement that any agent bidding high enough will lose and low enough will win, provided the other agents' bids are fixed. In the path selection setting, it could be that every path that contains a particular edge e contains other very expensive edges, so edge e would lose no matter how low she bids. However, if all of the edges along a path P bid sufficiently low, then path P should win. One way to look at this is that if the total cost of path P is sufficiently low compared to *every* other path (since every other path contains at least one edge not in P), then P should win. On the other hand, if there is at least one cheap path P and edge $e \notin P$, then by bidding sufficiently high, e should be able to force itself to lose. Second, since we are comparing the merits of various paths, it makes sense that raising the bids of edges not on path P should only make P more attractive. In particular, if P is the winner, it should still win even if other edges bid higher. We call this the *independence* property, because it implies that our comparison between paths P and Q is independent of the bids on all other edges. We summarize these properties below.

- (path autonomy) Given any bids b_{-P} for the edges off the path P, if all the edges on P bid sufficiently low then P will win.

- (edge autonomy) Suppose P is a path, edge $e \notin P$, and the bids along P are fixed at b_P. Then there exists some bid T such that if e bids T or higher, no path using e will win, no matter what the bids of the edges not in $P \cup \{e\}$.

- (independence) If path P wins, and an edge $e \notin P$ raises its bid, then P will still win.

Our last property, which we call *sensitivity*, is the most technical. It essentially says that if two paths are "tied," then perturbing the bids appropriately should break the tie.

Definition 3.4.1 *Suppose that for some set of bids, path P wins. Suppose that there exists an edge e such that arbitrarily small perturbations of e's bid cause another path Q to win. Then we say paths P and Q are* tied for the lead.[1]

Sensitivity just says that the mechanism should pay attention to all of the bids. If P wins but is tied with Q for the lead, then any increase in the bid of any edge in $P - Q$ or any decrease in the bid of any edge in $Q - P$ should make Q more attractive than P. Thus, any of these perturbations should make P lose. However, they may not make Q win, since there may be other paths also tied for the lead. The rigorous definition appears below.

- (sensitivity) Suppose path P wins, and Q is tied with P for the lead. Then increasing b_e for any $e \in P - Q$ or decreasing b_e for any $e \in Q - P$ causes P to lose.

We now record a simple consequence of the independence property to which we have already alluded.

[1]Note that, by independence, e belongs to either P or Q.

Proposition 3.4.2 *If lowering the bid on some edge causes the winning path to change, then the new winning path must contain that edge.*

Proof: Suppose P wins, but lowering the bid on edge e causes path Q to win, where $e \notin Q$. Raising e's bid back to its original value changes the winning path back to P, contradicting independence. ■

Now that we have defined independence and edge autonomy, we can state precisely what we mean when we say that an edge e "bids ∞." We mean that we are about to consider several settings for the bids along particular paths in the graph. For each of these settings, the edge autonomy property guarantees there is a threshold such that if b_e is above the threshold, e loses. Thus, if we set b_e anywhere above the maximum of these thresholds, e loses for each of these settings. By the independence property, the precise value of b_e does not matter, as it does not affect which path wins. To avoid running through this argument every time, we simply write that "e bids ∞."

3.5 Min function mechanisms

We define a broad class of truthful mechanisms called min function mechanisms, that allow us to easily introduce a bias toward certain paths, say ones with fewer edges. For each $s_1 - s_2$ path P in the graph, define a positive real valued function f_P of the vector of bids b_P. The mechanism evaluates each function $f_P(b_P)$ and selects the path with the lowest function value, breaking ties using any arbitrary rule that does not depend on the bids of edges not involved in the tie. (Since there are possibly exponentially many paths, we cannot necessarily compute these mechanisms in polynomial time, even if the functions are simple. But this is of no concern to us at present, since we are after a characterization result.) In order

for the mechanism to be truthful and satisfy the desired properties, we require the functions to satisfy the following:

- $f_P(b_P)$ is continuous and strictly increasing in b_e, for each $e \in P$
- (infinite limit) $\lim_{b_e \to \infty} f_P(b_P) = \infty$ for each $e \in P$
- (zero limit) $\lim_{b_P \to 0} f_P(b_P) = 0$.

Notice that the VCG shortest path mechanism is a special case, where the function f_P simply sums the bids along the path P: $f_P(b_P) = \sum_{e \in P} b_e$. Another example of a min function mechanism is to set $f_P(b_P) = \ln(1 + |P|) \sum_{e \in P} b_e$. This would introduce a mild bias against paths with many edges. Another example would be $f_P(b_P) = \prod_{e \in P} b_e^2$, although it is not clear why this one would be useful.

Theorem 3.5.1 *The min function path selection rule yields a truthful mechanism.*

Proof: We must show that this path selection rule is monotone, i.e., a losing edge can never cause itself to win by raising its bid. If P is currently the winning path and $e \notin P$, then $f_P(b_P)$ is the minimum function value. Raising b_e only increases the function values for all paths including e, but does not affect f_P, so path P still wins. Thus, the mechanism is monotone.

We now show that every edge in the winning path has a threshold bid. Let P be the path not including e whose function value is minimum. (It exists because the graph is 2-edge connected.) Let T be the largest bid for e such that some path Q containing e has $f_Q(b_{Q-e}, T) \leq f_P(b_P)$. As e raises its bid towards T, the winning path could change, but always contains e. Once b_e is raised beyond T, path P wins. ■

Notice that our proof relies only on the functions f_P increasing with infinite limit. Continuity and the zero limit are not important for truthfulness, and we can dispense with the strictness if we employ an appropriate tie breaking rule (such as numbering the paths and breaking ties by the lowest numbered path). We included these extra requirements on the f_P's so that min function mechanisms will satisfy our desired properties.

Theorem 3.5.2 *Min function mechanisms satisfy the edge and path autonomy, independence and sensitivity properties.*

Proof: Path autonomy follows from the zero limit property, since the functions are positive valued. Edge autonomy follows from the infinite limit property and the functions being increasing. Independence follows because the f_P's are strictly increasing and unaffected by the edges not on P, and the tie-breaking does not depend on edges not involved in the tie. Sensitivity follows because the f_P's are continuous and *strictly* increasing. ■

Theorems 3.5.1 and 3.5.2 show that on any graph, a min function mechanism is truthful and satisfies the desired properties. We now show a partial converse, namely that on two large classes of graphs, every truthful mechanism satisfying the desired properties is a min function mechanism. The first class is all graphs that contain an $s_1 - s_2$ arc. The second class is all graphs consisting of a connected graph containing s_1 and s_2, plus two parallel $s_1 - s_2$ paths whose nodes are disjoint from each other and from the rest of the graph.

We would like to have some concept of the bids on one path being "less than" the bids on another path. As we have noted, the independence property implies that, when deciding the winner between paths P and Q, the bids on the other edges are irrelevant. Thus, we might as well assume that the other edges all bid

∞. Suppose P bids b_P, Q bids b_Q, and all other edges bid ∞. If P wins, then we will write $b_P \prec b_Q$. For this to be defined, it is necessary that the bids b_P and b_Q can be *realized simultaneously*, i.e., they agree on the common edges $P \cap Q$. Note that if P and Q share a node (other than s_1 and s_2) then we can put together pieces of P and Q to create another finite path L. If L is the winner, then we cannot necessarily compare b_P and b_Q. In this case, we say that b_P and b_Q are *incomparable*. If P and Q are node-disjoint then they are the only two finite paths, so one of them must win and the bids are comparable. As a matter of notation, if v is an explicit vector that we intend to interpret as a vector of bids for path P, we will write it as $(v)_P$.

Lemma 3.5.3 *Given a mechanism satisfying the independence property, suppose P, Q, and L are three $s_1 - s_2$ paths, the bids b_P and b_L can be realized simultaneously and are comparable, and Q is node-disjoint from P and L. If $b_P \prec b_Q$ and $b_Q \prec b_L$, then $b_P \prec b_L$.*

Proof: Suppose on the contrary that when P bids b_P, L bids b_L, and the other edges all bid ∞, path L wins. Now lower the bids on Q to b_Q. By Proposition 3.4.2, either L still wins, or some path containing an edge in Q wins. But since Q is node-disjoint from P and L, every such path besides Q contains an edge that bid ∞. Thus, either L or Q wins. If L wins, then independence implies it still wins after we raise the bids on all edges in $P - L$ to ∞, but this contradicts the assumption $b_Q \prec b_L$. If Q wins, then it still wins after we raise all the bids on $L - P$ to ∞, which contradicts $b_P \prec b_Q$. ■

Theorem 3.5.4 *If graph G contains the edge $s_1 - s_2$, then any truthful mechanism for the $s_1 - s_2$ frugal path problem on G that satisfies the independence, sensitivity and edge and path autonomy properties is a min function mechanism.*

Proof: For each path P with bids b_P, we must first define a value for $f_P(b_P)$. Let R denote the arc from source to sink. We call this the *reference* arc. Since R is node-disjoint from every other $s_1 - s_2$ path, $(c)_R$ is comparable to every bid for every other path. Define $f_R((c)_R) = c$. For each other path P and set of bids b_P, define

$$f_P(b_P) = \sup\{c : (c)_R \prec b_P\}. \tag{3.1}$$

In other words, $f_P(b_P)$ is defined to be the threshold bid for the reference arc when P bids b_P and all other edges bid ∞. Since the mechanism is truthful, R will win if it bids below this threshold, and lose if it bids above the threshold.

Now we must show that the functions f_P select the correct path. Suppose on the contrary that $f_P(b_P) < f_Q(b_Q)$, but the mechanism selects path Q. By independence, if we raise all other bids to ∞, then Q still wins, so $b_Q \prec b_P$. By definition of f_P and f_Q, neither P nor Q can be R. Now let $c \in (f_P(b_P), f_Q(b_Q))$, so $b_P \prec (c)_R$ and $(c)_R \prec b_Q$. Applying Lemma 3.5.3 yields $b_P \prec b_Q$, which is a contradiction.

Finally, we must prove that the functions f_P satisfy the three properties we require of the functions defining a min function mechanism. Edge autonomy implies that $f_P(b_P) \to \infty$ as any bid along P goes to ∞, and path autonomy implies that $f_P(b_P) \to 0$ as $b_P \to 0$, since no matter what the reference arc bids, P must be able to lose to or beat it. Truthfulness implies the functions f_P are monotone increasing, since raising b_e can never cause a path containing e to beat the reference arc. Strict monotonicity and continuity are obvious for f_R, but are a bit more involved for the other paths.

To show the f_P's are strictly increasing, we fix some path P, some $e \in P$, and fix b_P to some bid $\bar{b}_P$. Set R's bid to $f_P(\bar{b}_P)$, and all other bids to ∞. Suppose

that R wins (a similar argument holds when P wins). Then any increase to b_R causes P to win. Thus any decrease to b_e causes P to win (by sensitivity). Let us decrease b_e by ϵ, so that $b_P = (\bar{b}_{P-e}, \bar{b}_e - \epsilon)$ and P wins. We want to show that the threshold bid for R is lower, which shows that $f_P(b_P)$ is lower. If not, than any decrease to R's bid causes R to win. So by sensitivity, any increase in b_e causes R to win. In particular, raising b_e by $\frac{\epsilon}{2}$ so that $b_P = (\bar{b}_{P-e}, \bar{b}_e - \frac{\epsilon}{2})$ causes R to win. But we already argued that P wins in this situation, so we've reached a contradiction.

To show continuity, we again argue by contradiction. Suppose we have some path P such that f_P is not continuous. Then f_P is discontinuous in some coordinate b_e, where e is an edge of P. Since f_P is monotone increasing in b_e, there must be some jump discontinuity at some point $b_P = \bar{b}_P$, where $f_P(\bar{b}_{P-e}, \bar{b}_e^-) = x$, $f_P(\bar{b}_{P-e}, \bar{b}_e^+) = y$, $x < y$, and the $^+$ and $^-$ denote limits as $b_e \to \bar{b}_e$ from above and below. Suppose $f_P(\bar{b}_P) = x$ (a similar argument holds when $f_P(\bar{b}_P) = y$). Suppose R bids $\frac{x+y}{2}$ and P bids $\bar{b}_P$, so P wins. Any increase in e's bid would raise $f_P(b_P)$ above y, so R would win. Thus, by sensitivity, any decrease in R's bid causes R to win, in particular for $b_R \in (x, \frac{x+y}{2})$, which is a contradiction. ■

Theorem 3.5.5 *If the graph G consists of some connected graph including nodes s_1 and s_2, plus two extra $s_1 - s_2$ paths that are disjoint from the rest of graph, then any truthful mechanism for the $s_1 - s_2$ frugal path problem on G that satisfies the independence, sensitivity and edge and path autonomy properties is a min function mechanism.*

Proof: Let $\mathcal{M}$ denote the mechanism. Let us call one of the disjoint paths R (for *reference* path) and the other disjoint path S (for *secondary* reference path). Define $f_R((c \cdot \mathbf{1})_R) = c$, where $\mathbf{1}$ denotes the vector of all ones, i.e., every edge on

R bids c. For any $P \neq R$, define

$$f_P(b_P) = \sup\{c : (c \cdot \mathbf{1})_R \prec b_P\}.$$

We also need to define $f_R(b_R)$ when the bids are not all the same, but let us postpone this for now. The proof that the functions f_P for $P \neq R$ are strictly increasing, continuous, and have the two limit properties, is exactly the same as in the proof of Theorem 3.5.4, except that every time we set the reference arc bid to c in that argument, we set it to $c \cdot \mathbf{1}$ here. Therefore, the range of each function f_P (for $P \neq R$) is $(0, \infty)$.

We want to show that the mechanism $\mathcal{M}$ always selects a path of minimum function value. In other words, if $f_P(b_P) < f_Q(b_Q)$, we need to rule out the possibility that Q wins. If Q were to win, then, by the independence property, it would still win after we raise the bids of all edges not in P or Q to ∞, so we would have $b_Q \prec b_P$. Thus, it suffices to rule out this possibility. In other words, we want to show that if $f_P(b_P) < f_Q(b_Q)$ and b_P and b_Q are comparable, then $b_P \prec b_Q$. There are several cases, depending on whether P or Q is one of the two reference paths.

Case 1: $f_P(b_P) < f_Q(b_Q)$ and neither P nor Q is R. We note that if $b_R = c \cdot \mathbf{1}$ where $c \in (f_P(b_P), f_Q(b_Q))$, then $b_P \prec b_R \prec b_Q$ so we can apply Lemma 3.5.3 to obtain $b_P \prec b_Q$.

Before we can consider the other cases, we must define $f_R(b_R)$ when the bids along R are not all the same. Let

$$f_R(b_R) = \sup\{c : \exists b_S \text{ s.t. } f_S(b_S) = c \text{ and } b_S \prec b_R\}.$$

Now that we have shown the other functions f_P to have the properties we want, we can use them to prove the properties for f_R.

Case 2: $f_P(b_P) < f_R(b_R)$ **and** $P \neq S$**.** By definition of f_R, there exist bids $\bar{b}_S$ for path S such that $\bar{b}_S \prec b_R$ and $f_S(\bar{b}_S) \in (f_P(b_P), f_R(b_R)]$. By Case 1 we have $b_P \prec \bar{b}_S$. Applying Lemma 3.5.3 gives $b_P \prec b_R$. The argument is similar when $f_P(b_P) > f_R(b_R)$.

Case 3: $f_S(b_S) < f_R(b_R)$**.** Select any other $s_1 - s_2$ path P in the graph. Since the range of f_P is $(0, \infty)$, there exists a bid $\bar{b}_P$ such that $f_P(\bar{b}_P) \in (f_S(b_S), f_R(b_R))$. By Case 1 we have $b_S \prec \bar{b}_P$, and by Case 2 we have $\bar{b}_P \prec b_R$, so by Lemma 3.5.3 we have $b_S \prec b_R$. The argument is similar when $f_S(b_S) > f_R(b_R)$.

For $P \neq R$, we already know that f_P is strictly increasing, continuous, and has the two limit properties. Now that we know the mechanism selects a path of smallest function value as the winner, we can prove that f_R also has these properties, using arguments similar to the proof of Theorem 3.5.4. Thus, our functions satisfy the required properties to define a min function mechanism, and $\mathcal{M}$ always selects a path of minimum function value. Since a min function mechanism may break ties arbitrarily, we choose ours to break ties the same way that $\mathcal{M}$ does. Thus, we have successfully represented $\mathcal{M}$ as a min function mechanism. ■

We conjecture that Theorem 3.5.5 extends to all graphs containing 3 node-disjoint $s_1 - s_2$ paths P, Q, and R. In such a graph, when all other edges incident to an internal node of P or Q bid ∞, the mechanism reduces to a case covered by Theorem 3.5.5. It seems that one ought to be able to extend the min function representation to the whole graph.

3.6 Costly example for min function mechanisms

We show here that any min function mechanism can be forced to pay $L(1+k\epsilon)$, where L is the cost of and k is the number of edges on the winning path, even if there is some node-disjoint path of cost $L(1+\epsilon)$. If $k\epsilon$ is large, then we would pay a large factor times the actual cost of the chosen path, even though we are in a "tight market." Moreover, for fixed ϵ, we pay $\Theta(k)$ times the cost of the *longer* path. This contrasts with the Vickrey auction, where we only pay as much as the 2nd lowest bid.

Consider a min function mechanism on a graph consisting of two node-disjoint s_1-s_2 paths, P and Q. Fix some $L, \epsilon \in \mathbb{R}^+$, and let $|P|$ and $|Q|$ denote the number of edges in P and Q. Let b_P^i denote the vector of bids along P where all edges bid $\frac{L}{|P|}$, except for the ith edge, which bids $\frac{L}{|P|}+\epsilon L$. Similarly, let b_Q^i denote the bids along Q where all edges bid $\frac{L}{|Q|}$, except for the ith edge, which bids $\frac{L}{|Q|}+\epsilon L$. Suppose without loss of generality that $f_Q(b_Q^1)$ is the maximum of the function values $f_P(b_P^1),\ldots,f_P(b_P^{|P|}), f_Q(b_Q^1),\ldots,f_Q(b_Q^{|Q|})$. Let b_P^0 denote the vector of bids when every edge along P bids $\frac{L}{|P|}$. Now consider how the mechanism acts when P bids b_P^0 and Q bids b_Q^1. Since $f_P(b_P^0) < f_P(b_P^1) \leq f_Q(b_Q^1)$, path P wins. To determine the payments, we must figure out the threshold bid for each edge in P. Since, for $i=1,\ldots,|P|$, $f_P(b_P^i) \leq f_Q(b_Q^1)$, P still wins (or ties) even if one of the edges along P raises its bid by ϵL. Thus, the threshold bid for each edge in P is at least $\frac{L}{|P|}+\epsilon L$, so the total amount paid by the mechanism is at least $L(1+|P|\epsilon)$, even though the length of P is L and Q is only $L(1+\epsilon)$. If the graph consists of more than just P and Q, we can achieve the same result by setting the bids of all other edges to be ∞.

Theorem 3.6.1 *Any truthful mechanism on a graph that contains either an* s_1-s_2

arc or three node-disjoint $s_1 - s_2$ paths and satisfies the independence, sensitivity and edge and path autonomy properties can be forced to pay $L(1 + k\epsilon)$, where the winning path has k edges and costs L, even if there is some node-disjoint path of cost $L(1 + \epsilon)$.

Proof: If G belongs to one of the two classes covered by Theorems 3.5.4 and 3.5.5, then the mechanism is a min function mechanism. Otherwise, select three node-disjoint paths, P, Q, and R. When all of the other edges of G bid ∞, we can essentially ignore them, by the independence property. We can then apply Theorem 3.5.5 to the remaining graph, which consists of P, Q, and R. Thus, the mechanism reduces in this case to a min function mechanism. The preceding discussion shows how any min function mechanism can be forced to pay the specified amount. ■

3.7 An exceptional mechanism

In this section, we exhibit a truthful mechanism (for a graph outside our two classes) that satisfies the independence, sensitivity and edge and path autonomy properties but is not a min function mechanism. Let the graph G be a 4-node graph consisting of 2 $s_1 - s_2$ paths P and Q, each having two edges. Let x_1 and x_2 be the bids along P, and y_1 and y_2 be the bids along Q. Let us describe the winning path via a "phase diagram." For a given (y_1, y_2), draw a dividing line in the $x_1 - x_2$ plane with x_1-intercept $(y_1 + \frac{y_2}{2}, 0)$ and x_2-intercept $(0, \frac{y_1}{2} + y_2)$. If (x_1, x_2) lies on or below the line, then P wins. Otherwise Q wins. In Figure 3.1, P loses if it bids x^1, wins if it bids x^2, and ties but wins the tiebreaker if it bids x^3.

It is easy to check that this mechanism is truthful and satisfies the desired properties. For P, the threshold bids are where the horizontal and vertical lines

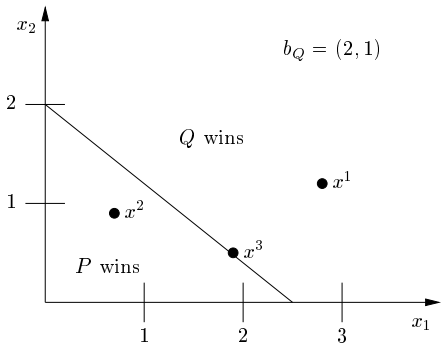

Figure 3.1: The winning path is determined by this "phase diagram."

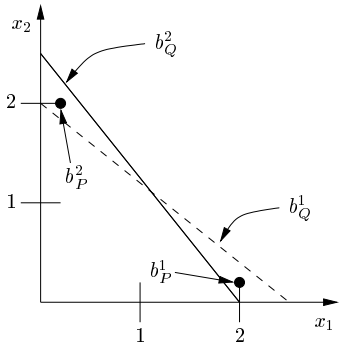

Figure 3.2: This figure shows that the $\prec$ relation on bids is non-transitive.

through (x_1, x_2) cross the dividing line in the phase diagram, provided this crossing occurs in the first quadrant, and zero otherwise. Increasing either bid along Q pushes the dividing line both up and to the right in the phase diagram, which cannot cause Q to win unless it was already the winner. Since increasing either bid can push the dividing line out arbitrarily far, the edges of Q have threshold bids as well. Thus, the mechanism is truthful. Since, for fixed (y_1, y_2), the region of the phase diagram where P wins contains a neighborhood of the origin, P can win by bidding low enough. Path Q can also win by bidding sufficiently low, since this moves the dividing line arbitrarily close to the origin. Hence path autonomy holds. With b_Q fixed, either edge on P can cause itself to lose by bidding larger than the appropriate intercept of the dividing line defined by b_Q. With b_P fixed, either edge on Q can cause itself to lose, because both intercepts of the dividing line go to ∞ as either edge's bid does. Thus, edge autonomy holds. Independence holds trivially because there are only two paths from which to choose. The two paths are tied for the lead exactly when (x_1, x_2) is on the dividing line. In this case P wins, but increasing either bid on P causes the point to move above the line so Q wins, and decreasing either bid on Q causes the line to move down and left, so Q wins. Thus, sensitivity holds.

To show that this mechanism cannot be represented by a min function mechanism, consider the bids $b_P^1 = (2, \frac{1}{5})$, $b_P^2 = (\frac{1}{5}, 2)$, $b_Q^1 = (2, 1)$ and $b_Q^2 = (1, 2)$. Figure 3.2 overlays the phase diagrams. The dashed dividing line corresponds to Q's bid b_Q^1 and the solid line to b_Q^2. From the phase diagram, we see that $b_P^1 \prec b_Q^1 \prec b_P^2 \prec b_Q^2 \prec b_P^1$, and perturbing any of these bids a small amount would not change the outcome. If the mechanism were representable by a min function mechanism, then we would have $f_P(b_P^1) < f_Q(b_Q^1) < f_P(b_P^2) < f_Q(b_Q^2) < f_P(b_P^1)$,

which is a contradiction.

This example is easily extended to two node-disjoint $s_1 - s_2$ paths, each with at least 2 edges. If the bids on P are $x_1, \ldots, x_k$ and the bids on Q are $y_1, \ldots, y_l$, then in the phase diagram we represent b_Q by the dividing line with intercepts $y_1 + \frac{y_2}{2} + \sum_{i \geq 3} y_i$ and $\frac{y_1}{2} + y_2 + \sum_{i \geq 3} y_i$, and we represent b_P by the point $(x_1 + \sum_{i \geq 3} x_i, x_2)$. Thus, the extra bids on Q just move the dividing line outward and the extra bids on P just move the point to the right.

3.8 Truthfulness without independence

The last section showed that some assumption on the structure of the graph is necessary to prove our characterization result. Here we show that the independence assumption is also necessary, by exhibiting a truthful mechanism that satisfies sensitivity and edge and path autonomy but violates independence.

Consider a graph consisting just of three parallel $s_1 - s_2$ arcs. If $b_1 + b_2 \leq 6b_3$, then the lower of b_1 and b_2 wins. Otherwise the lowest bid wins. Ties go to the lowest numbered bidder. This is clearly truthful and satisfies edge and path autonomy. Edges 1 and 2 are tied for the lead if $b_1 = b_2 \leq 3b_3$, because in this case edge 1 wins but any decrease to b_2 causes edge 2 to win. Since any increase in b_1 causes either b_2 or b_3 to win, the sensitivity property holds in this case. Edges 1 and 3 are tied for the lead if either ($b_1 + b_2 > 6b_3$ and $b_3 = b_1 \leq b_2$) or ($b_1 + b_2 = 6b_3$ and $b_3 \leq b_1 \leq b_2$), because in both cases b_1 wins, but any decrease in b_3 causes edge 3 to win. Sensitivity holds here, because any increase in b_1 causes either edge 2 or 3 to win. These are the only ways for edge 1 to be tied for the lead. A similar analysis holds for ties between edges 2 and 3. So the sensitivity property holds for this mechanism. If $b_1 = 2$, $b_2 = 3$ and $b_3 = 1$, then edge 1 wins, but if b_2 increases

to 5, then edge 3 wins. Thus, the independence property fails.

3.9 Efficient computation of min function mechanisms

To this point, we have not been concerned with how to compute min function mechanisms efficiently, since our goal was to prove a negative results. However, even though all min function mechanisms can be forced to overpay just as badly as VCG in the worst case, we have reason to believe that certain classes of them might perform better in practice. Here, we mention one attractive and natural class that can be computed in polynomial time.

Our original motivation for defining min function mechanisms was to provide a means for biasing the path selection rule against paths with a large number of hops. The hope is that such a path selection rule will choose a path with fewer hops, so we will pay bonuses to fewer edges. Here is one way to introduce such a bias. Let g be any positive, increasing (and easily computed) function defined on the positive integers, and define $f_P(b_P) = g(|P|)\sum_{e \in P} b_P$, where $|P|$ denotes the number of edges in path P.

This class of min function mechanisms can be computed in polynomial time, using $O(n)$ Bellman-Ford shortest path computations. A single run of Bellman-Ford computes the shortest path from s_1 to s_2 using k hops, for each $k = 1, \ldots, n-1$. To compute the winning path, we just multiply the length of the shortest k-hop path by $g(k)$, and take the minimum result. To compute the threshold bid for each winning edge e, we perform another Bellman-Ford shortest path computation on the graph with edge e deleted, to determine which path Q has the minimum function value among all paths not using edge e. There may be several values k for which the shortest k-hop path P_k uses edge e, and for which $f_{P_k}(b_{P_k}) \leq b_Q(b_Q)$.

For each of these, Q becomes preferred once e has raised its bid by $(b_Q(b_Q) - f_{P_k}(b_{P_k}))/g(k)$. Computing the maximum of these increments gives the bonus for edge e, because that is the lowest bid for which a path not containing e wins.

3.10 Characterization result for general team selection mechanisms

Min function mechanisms can be defined for the general problem of team selection in the obvious way: we just define a function f_P for every feasible team P. In the definitions of edge and path autonomy, independence and sensitivity, if we replace "edge" everywhere by "agent," and "path" by "team," we obtain the analogous properties for team selection mechanisms. In the proofs of Lemma 3.5.3 and Theorems 3.5.4 and 3.5.5, the only way in which we relied on the fact that the teams were paths is that no $s_1 - s_2$ path is a subset of another. Our definition of the team selection problem already specified that no feasible team should be a subset of another. Thus, analogs of these three results all hold for the general team selection problem. The analog of a graph containing the edge $s_1 - s_2$ is a set of feasible teams containing at least one team consisting of a single agent. The analog of a connected graph plus two extra $s_1 - s_2$ paths disjoint from the rest of the graph is a set of feasible teams containing at least three teams, where the agents composing two of the teams belong only to that team. Using these analogous concepts, we can prove analogs of Lemma 3.5.3 and Theorems 3.5.4 and 3.5.5. The proofs go through without modification. Theorem 3.6.1 did rely slightly on the structure of paths. The condition we need for an analog of this theorem to hold is that there are three feasible teams P, Q, and R such that the only feasible teams that are subsets of $P \cup Q \cup R$ are P, Q, and R.

3.11 Further work

It is natural to question how reasonable are our assumptions of independence, sensitivity and edge and path autonomy. At first glance, the independence assumption may appear to be the most innocent, but inspection of the proofs of Theorems 3.5.4 and 3.5.5 reveal that independence is the crux. Indeed, if we relax some of the assumptions on the functions defining min function mechanisms, requiring them only to be increasing with infinite limits, then we are still left with truthful mechanisms that seem intuitive. These can fail the sensitivity and path autonomy properties, but independence and edge autonomy are essential features. The first two properties, while natural, were assumed mostly to make the proof go through nicely. One open problem is to prove our characterization results (for the relaxed definition of min function mechanism) without assuming these two conditions.

The bigger open problem left by our work is to prove non-frugality of all truthful path selection mechanisms, without the independence assumption. Recently, after the initial publication of our work, Elkind et al. [19] indeed proved that every truthful mechanism (without any additional assumptions) can be forced to overpay just as badly as VCG in the worst case. They also examine mechanisms where they assume a commonly known probability distribution on edge costs, and they wish to find a mechanism for which truth-telling is a Bayesian Nash equilibrium. This is a much weaker solution concept than the concept of dominant strategies used throughout this dissertation, and hence it admits a larger class of mechanisms. Remarkably, they show that the mechanism in this class that minimizes the total expected payments (where the expectation is over the probability distributions on the edges) *is* a particular type of min function mechanism. In other words, even

though they are considering a wider class of admissible mechanisms, the optimal mechanism in this class is still truthful according to the stronger definition used throughout our work. This optimal mechanism is a min function mechanism whose definition is tuned for the given probability distributions. They also give examples of distributions where the expected total cost (to the agents) of the shortest path is $\Theta(n)$, as is the expected total payment by the optimal min function mechanism, but the VCG mechanism pays $\Theta(n\sqrt{n})$ in expectation.

Following our work, Talwar [67] investigated the behavior of the VCG mechanism on the general team selection problem. Recall the result of Bikhchandani et al. [11] that when the collection of feasible sets $\mathcal{F}$ are all of the spanning trees of a graph, the VCG mechanism never pays more than the cost of the cheapest feasible set that is disjoint from the one chosen. The main result of [67] is that applying the VCG mechanism to a team selection problem yields this worst-case guarantee if and only if the collection of feasible sets $\mathcal{F}$ is what they call a "frugoid," which is a generalization of a matroid. It would be interesting to determine whether there are set systems other than frugoids where some *other* mechanisms besides VCG can achieve a similar worst-case guarantee.

Feigenbaum et al. [21] ran some experiments to determine how much the VCG mechanism overpays on some artificial examples generated from the problem of routing between autonomous systems on the Internet. (We briefly discussed this model in Section 1.2.2.) They ran their experiments on a 5773-node graph where the nodes denote autonomous systems from a recent snapshot of the Internet, and the edges denote links between them. In their model, the nodes incur costs rather than the links. They assumed all node costs were 1, and computed VCG prices for randomly selected $s_1 - s_2$ pairs. They report that 68% of the node payments

were 1, 28% of them were 2, and the average node payment was 1.44. This means that VCG overpaid by 44% on average. However, we believe that this number is artificially small because of the design of the experiment. Since the node costs were all taken to be 1, computing the shortest path is equivalent to computing the path with the least number of hops. If deleting a winning node results in an alternate path using the same number of hops, then this node's threshold bid is 1, so she receives no bonus. As they report, this occurred 68% of the time. If we were to add some variation in the node costs, all of these zero bonuses would now become positive. It would be interesting to test how significantly this changes their experimental results. In this experiment, using min function mechanisms that are biased towards paths with few hops would not help, because VCG is already choosing the path with the fewest hops. In an experiment where the node costs vary widely, the shortest paths could often have more than the minimum number of hops. In these cases, it would be interesting to test whether min function mechanisms could give improved results.

Another open question involves the efficient computation of min function mechanisms. A naive implementation of the VCG shortest path mechanism requires one Dijkstra shortest path computation for each winning edge to compute its payment. However, Hershberger and Suri [33] show how to compute the entire mechanism in time dominated by two Dikstra computations. In Section 3.9 we showed how to compute a natural class of min function mechanisms using one Bellman-Ford shortest path computation per winning edge. It is an open problem whether one can compute this class of min function mechanisms using just $O(1)$ Bellman-Ford computations.

Chapter 4

Truthful discrete optimization

In this chapter, we show how to design truthful mechanisms for several combinatorial problems where the VCG mechanism does not apply because the objective function is not utilitarian. We want mechanisms that we can compute in polynomial time. Since some of the problems we consider are NP-hard, we design approximation algorithms for them. In all of the problems we consider, an agent's allocation causes her to incur a cost, not attain a benefit. Therefore, we refer to agent i's allocation q_i as her *load*. As we mentioned in Sections 1.3 and 2.4, t_i denotes i's cost per unit load, so $v_i = -t_i q_i$.

Our main tool is Theorem 2.4.3, which shows that a mechanism is truthful if and only if each agent i's load is monotone decreasing in her bid, and her payment is given by a particular integral of her load. Thus, the first order of business in creating an efficiently-computable truthful mechanism is to obtain a monotone allocation algorithm. In some cases, we are able to compute the payment integrals directly in polynomial time. In others, we must resort to the general tricks developed in Section 2.6.

We start off by looking at some load-balancing problems on uniformly related parallel machines. We describe this model here, in order to give an idea of how the agents' types are related to the data of the optimization problem. There are n jobs that we want to run on m machines. Each job j has a processing requirement p_j (the amount of load it represents), and each machine i runs at some speed s_i. If job j is scheduled on machine i, it takes p_j/s_i units of time to complete. The goal is to allocate the jobs to machines so that the last job finishes as soon as possible. Each machine is an agent, who incurs a cost proportional to the total

time she spends processing. Agent i's load q_i is just $\sum p_j$, summing over the jobs that are assigned to i, and we take her type t_i to be $1/s_i$, so that $v_i = -t_i q_i$. We examine three different load balancing problems in this model of uniformly related machines.

We first consider scheduling to minimize makespan, the problem usually denoted by $Q||C_{\max}$ in the literature. The objective is to allocate the jobs such that the last job finishes as early as possible. This problem is NP-hard, and the standard approximation algorithms (greedy load-balancing or the PTAS[1] [35]) cannot be used in truthful mechanisms, because they are not monotone. We give a 2-approximation mechanism that is truthful in expectation, based on randomized rounding of the optimal fractional solution. In this case, the truthfulness alone is not the barrier to approximability – the optimal allocation is monotone, so if we had unbounded computational resources, we could obtain a truthful mechanism that yields the optimal solution. We also prove that our mechanism is frugal, in contrast to the results of Chapter 3 for the path selection problem.

We next consider scheduling with preemption to minimize makespan, which is usually denoted $Q|\text{pmtn}|C_{\max}$ in the literature. In this model, we may start a job on one machine, interrupt it, and finish it later on another machine. There are multiple algorithms that solve this problem in polynomial time [29, 37], but not all of them are monotone. We give a method for converting any optimal algorithm into a monotone one, thereby yielding an optimal truthful mechanism.

For $Q||\sum C_j$, the problem of scheduling uniformly related parallel machines to minimize the sum of completion times (or equivalently, the average completion time), we observe that the standard algorithm is already monotone. For both this

[1] A PTAS is a family of algorithms that, for fixed ϵ yields a $1+\epsilon$ approximation in polynomial time.

problem and the previous two, we show how to compute the payments explicitly in polynomial time.

We show that optimizing an affine function over a fixed set can be done truthfully, if the linear decision variables are the loads and their coefficients are monotone functions of the corresponding bids. The feasible set is not allowed to depend on the bids. Problems that fall under this category include some linear programming problems, minimum cost flow, and uncapacitated facility location.

There are special cases of the uncapacitated facility location problem that can be solved in polynomial time. However, the general problem is NP-hard. For the problem defined on general metric spaces, we adapt the "online" algorithm of [48] to give a truthful approximation mechanism with constant performance guarantee, provided the facility costs are known to come from a constant interval.

Even though maximum flow can be solved via linear programming, it is not covered by our general result above because we take the edge capacities to be the agent types, so the feasible set of flows depends on the bids. Instead, we show how to attain a monotone algorithm via combinatorial methods. Interestingly, no optimal maximum flow mechanism that satisfies the voluntary participation condition can obtain the optimal flow (although it can be made to come arbitrarily close).

For one problem, we also derive a lower bound on the approximation ratio achievable by a truthful mechanism. We consider scheduling on uniformly related parallel machines to minimize the weighted sum of completion times ($Q||\sum w_j C_j$). In contrast to the simple optimal algorithm for unweighted completion times, we can show for this problem that *no* truthful mechanism can achieve an approximation ratio better than $\frac{2}{\sqrt{3}}$, even using unbounded computation.

Throughout this chapter, the types t_i represent both data for the optimization problem, and part of the formula expressing i's valuation function. In the discussion of our load balancing model, we said that a machine's cost is proportional to her processing time. Since t_i is the inverse of i's speed, then its units are time per unit work. We said the valuation $v_i = -t_i q_i$. Since the valuation is in units of money and $t_i q_i$ is in units of time, technically there should be a constant of proportionality on the right hand side. For simplicity of presentation, we choose our unit of currency so that the constant of proportionality is one. Everything we present still works if we let the constant vary from machine to machine, so long as the constants are known to the mechanism (not part of the private data).

4.1 Related work

Compared to the amount of work that has addressed computational issues surrounding the VCG mechanism, considerably less has addressed non-utilitarian objective functions. Nisan and Ronen [58] applied the mechanism design framework to some standard optimization problems in computer science, suggesting general objective functions. This chapter is closest in spirit to their work.

While our main example is the problem $Q||C_{\max}$, scheduling on machines with speeds, the main focus in [58] is a similar NP-hard problem $R||C_{\max}$, scheduling on unrelated machines. In that problem, each machine has n items of private data, the amounts of time it would take for it to process each job (so the one-parameter monotonicity result of Theorem 2.4.3 does not apply). The output is an allocation of jobs to machines, and the cost to a machine equals the total time it spends processing its jobs. The best approximation algorithm for this problem yields a performance guarantee of 2 [42], but nobody has been able to

convert this into a truthful mechanism. Nisan and Ronen provide a simple truthful mechanism (consisting of a separate Vickrey auction for each job) that yields an m-approximation, where $m = |\mathcal{N}|$. They conjecture that *no* truthful mechanism has a better approximation guarantee, although the best lower bound they prove is 2. With strong additional restrictions on the types of payment schemes allowed, they prove a lower bound of m. Note the large gap between the best approximation factors known for a polynomial-time algorithm, 2, and for a truthful mechanism, m. We have a similar gap for $Q||C_{\max}$ between the PTAS of [35] and the truthful 2-approximation of Theorem 4.2.4.

The lower bound of 2 for $R||C_{\max}$ stands in contrast to our truthful (but not polynomial-time) mechanism that exactly solves $Q||C_{\max}$. Ronen and Nisan do give a mechanism that solves $R||C_{\max}$ exactly, but only in a much stronger model in which the mechanism is allowed to observe the machines process their jobs and compute the payments *afterwards*, which makes it easy to penalize lying agents.

4.2 Load balancing

We consider the problem $Q||C_{\max}$, which we mentioned in the introduction. This problem is NP-hard, although there is a PTAS [35]. We are given n jobs and m machines. The jobs represent amounts of work $p_1 \geq \ldots \geq p_n$. The output is an assignment of jobs to machines. Machine i runs at some speed s_i, so it must spend $\frac{p_j}{s_i}$ units of time processing each job j assigned to it. The load on machine i is $q_i(b) = \sum p_j$, where the sum runs over jobs j assigned to i. Each machine is an agent, who incurs a cost equal to the time she spends processing her jobs. We take the type to be $t_i = \frac{1}{s_i}$ so that the machines' costs are of the correct form, $v_i = -t_i q_i(b)$. The mechanism's goal is to minimize the completion time of the last

job on the last machine, i.e., $g(O(b), t) = C_{\max} = \max_i t_i q_i(b)$.

The mechanism design problem for $Q||C_{\max}$ contrasts sharply with the mostly negative results of Nisan and Ronen [58] (see Section 4.1). We show that truthfulness alone does not prohibit achieving the optimal allocation. Then we give a randomized polynomial-time truthful mechanism that yields a 2-approximation for $C_{\max}$.

Definition 4.2.1 *A vector* $(q_1, \ldots, q_m)$ *is smaller than* $(\bar{q}_1, \ldots, \bar{q}_m)$ lexicographically *if, for some* i, $q_i < \bar{q}_i$ *and* $q_k = \bar{q}_k$ *for all* $k < i$.

Proposition 4.2.2 *There is a truthful mechanism (not polynomial-time) that outputs an optimal solution for* $Q||C_{\max}$ *and satisfies the voluntary participation property.*

Proof: Among the optimal allocations of jobs, our algorithm selects the one in which the load vector $(q_1, \ldots, q_m)$ is lexicographically minimum. Clearly, a machine raising its bid b_i (i.e., announcing it is slower) will not cause the allocation to change unless that machine is the bottleneck. In this case raising b_i will either result in the same optimal allocation, or cause machine i to get less load. Thus, the loads q_i are decreasing, so by Theorem 2.4.3 this output algorithm admits a truthful payment scheme given by (2.5). As we just argued, $q_i(b_{-i}, \cdot)$ is constant except for jumps at the breakpoints where machine i becomes the bottleneck and the optimal solution shifts load off of i, so $q_i(b_{-i}, 0)$ is a decreasing step function. Moreover, a sufficiently slow machine receives no work, so $\int_0^\infty q_i(b_{-i}, x)dx < \infty$, and by Theorem 2.4.4 we can choose P to satisfy the voluntary participation property. ■

We now move to polynomial-time mechanisms. We cannot simply use any existing approximation algorithm because the work assigned to agent i typically

changes in complicated ways as her bid b_i changes. In particular, the PTAS in [35] relies on dynamic programming and rounding the job sizes down to the nearest power of $(1+\epsilon)$. Even if the dynamic programming can be done so that the load on a machine according to the rounded job sizes is decreasing in her bid, the rounding still causes problems. To see this, suppose that a machine were to announce a slightly slower speed, causing her to receive a different set of jobs, and the load according to the rounded job sizes decreased by less than a $(1+\epsilon)$ factor. Suppose the actual sizes of the old set of jobs were already powers of $(1+\epsilon)$, so that the rounding had no effect, but the actual sizes of the new set of jobs were all just below powers of $(1+\epsilon)$, so that all the rounded sizes were smaller by a factor of almost $(1+\epsilon)$. Then the total actual size of the new set of jobs is larger than the actual size of the old set of jobs. Thus, the overall algorithm is not monotone, even though it is with respect to the rounded job sizes.

The following greedy load balancing algorithm also fails to be monotone. Consider the jobs one at a time, from largest to smallest. Start off with all of the machines empty. When considering job j, place it on the machine that would have the smallest overall processing time, if j were added to it. One can easily show that this algorithm gives a 2-approximation. However, the following example shows that it fails to be monotone. Consider scheduling three jobs with processing requirements $p_1 = 2$ and $p_2 = p_3 = 1+\epsilon$ on two machines of speeds $s_1 = 1+\epsilon$ and $s_2 = 1$. First job 1 is assigned to machine 1, then jobs 2 and 3 both go on machine 2. Now suppose that machine 1's speed decreases to $1-\epsilon$. In this case, the algorithm would put job 1 on machine 2 and jobs 2 and 3 on machine 1. By decreasing its speed (increasing its bid), machine 1 caused its load to increase from 2 to $2+\epsilon$.

Having shown why known approximation algorithms are not monotone, we now consider how to create an algorithm that is monotone. We want to make sure that i's load never increases if her bid increases (i.e., announced speed decreases), while the other machines' bids stay fixed. In creating an approximation algorithm, our first task is to establish a lower bound on the cost of an optimal solution.

We first note that our problem is equivalent to bin packing with uneven bins, which leads to a lower bound on $C^*_{\max}$, the optimal makespan. This bound is implicit in the $\frac{3}{2}$-approximation algorithm of [35]. Given a guess T at the value of $C^*_{\max}$, we create a bin of size T/b_i for each machine i. The size of a machine's bin is the maximum load we can assign to it if the machine is to finish all its jobs by time T. Then $C^*_{\max} \leq T$ if and only if there exists an assignment of jobs to bins such that each bin is at least as large as the total size of all the jobs assigned to it.

For each j, let us consider the j largest jobs. These will give us a lower bound T_j on $C^*_{\max}$. Let us number both the bins and jobs from largest to smallest, so $b_1 \leq \ldots \leq b_m$ and $p_1 \geq \ldots \geq p_n$. Given our guess for T, let $i(j,T)$ be the smallest bin that is large enough to accommodate job j, all by itself, i.e., $i(j,T) = \max\{i : T/b_i \geq p_j\}$. Then either jobs $p_1, \ldots, p_j$ can all fit in the first $i(j)$ bins, or else $T < C^*_{\max}$. A necessary condition for the jobs to fit is that the total size of these jobs does not exceed the total size of the bins, i.e.,

$$T \geq \frac{\sum_{k=1}^{j} p_k}{\sum_{\ell=1}^{i(j,T)} \frac{1}{b_\ell}}. \tag{4.1}$$

Let us define

$$T_j = \min_i \max \left\{ b_i p_j, \frac{\sum_{k=1}^{j} p_k}{\sum_{\ell=1}^{i} \frac{1}{b_\ell}} \right\}. \tag{4.2}$$

We claim that T_j is a valid lower bound on $C^*_{\max}$, and it is the best lower bound we can obtain from this argument, using the j largest jobs. Notice that the first term in the max is increasing with i, whereas the second term is strictly decreasing

with i. Thus, the value of i that defines T_j in the minimization in (4.2) (call it i_j) is either the last i such that the second term in the max is larger (call this case 1), or the first i such that the first term in the max is larger (call this case 2). In either case, when $T = T_j$, the bin i_j is large enough to accommodate the job j, while the bin $i_j + 1$ is not. That is, $i(j, T_j) = i_j$. Moreover, the first i_j bins do have enough collective capacity to hold the j longest jobs. Thus, T_j satisfies the condition (4.1), so we cannot conclude $T_j < C^*_{\max}$. However, for every $T < T_j$, we claim that condition (4.1) fails, from which we can conclude that $T_j \leq C^*_{\max}$. Since $T < T_j$, clearly $i(j, T) \leq i_j$. Thus, in case 1 we have

$$T < T_j = \frac{\sum_{k=1}^{j} p_k}{\sum_{\ell=1}^{i_j} \frac{1}{b_\ell}} \leq \frac{\sum_{k=1}^{j} p_k}{\sum_{\ell=1}^{i(j,T)} \frac{1}{b_\ell}},$$

so condition (4.1) fails. In words, the bottleneck defining T_j is that the total capacity of the largest i_j bins is just enough to fit the top j jobs, so we cannot make them any smaller. In case 2, we have $i(j, T) < i_j$. Thus,

$$T < T_j = b_{i_j} p_j \leq \frac{\sum_{k=1}^{j} p_k}{\sum_{\ell=1}^{i_j - 1} \frac{1}{b_\ell}} \leq \frac{\sum_{k=1}^{j} p_k}{\sum_{\ell=1}^{i(j,T)} \frac{1}{b_\ell}}$$

so condition (4.1) fails in this case as well. In words, for T_j, the total capacity of the first $i_j - 1$ bins is not enough to fit the top j jobs, so the size of bin i_j causes the bottleneck, and we cannot make it any smaller. Thus, $T_j \leq C^*_{\max}$.

Let us define

$$T_{LB} = \max_j T_j = \max_j \min_i \max \left\{ b_i p_j, \frac{\sum_{k=1}^{j} p_k}{\sum_{l=1}^{i} \frac{1}{b_l}} \right\}. \tag{4.3}$$

Since $T_j \leq C^*_{\max}$ for all j, and T_{LB} is just the max of these lower bounds, T_{LB} is also a valid lower bound for $C^*_{\max}$.

We now consider how to use the T_{LB} lower bound to create an assignment of jobs to bins that is close to optimal. We first relax our requirements by allowing

fractional assignments. A *fractional assignment* of jobs to bins consists of a partition of each job j into pieces whose sizes sum to p_j and an assignment of these pieces to the bins. A fractional assignment is *valid* if each bin is at least as large as the total size of all fractional jobs assigned to it, and every bin receiving a piece of a job is large enough to contain that entire job (if it didn't have anything else in it).

We claim that if we set the bin sizes according to T_{LB}, then a valid fractional assignment exists, and can be constructed greedily. Assign jobs $1, 2, \ldots, (k-1)$ to bin 1, where k is the first job that would cause the bin to overflow. Then assign to bin 1 a piece of job k exactly as large as the remaining capacity in bin 1. Continue by assigning jobs to bin 2, starting with the rest of job k, and so on. We now show that this assignment is valid. Since $T \geq T_n$, the total bin size is at least as large as the total size of all jobs, so all jobs are fully assigned. By construction, no bin is overfilled. It remains to show that, for each j, job j is fractionally assigned only to bins of size at least p_j. If the bins were sized according to T_j, then the total size of all bins at least as large as p_j is at least $\sum_{k=1}^{j} p_k$, by definition of T_j, so the greedy assignment would finish assigning job j before it reached the bins of size $< p_j$. In fact, T_j is the smallest value of T for which this is true. Since $T_{LB} \geq T_j$, job j is fully assigned at least as early when the bins are sized according to T_{LB}. Thus, the greedy fractional assignment is valid. The lemma below follows.

Lemma 4.2.3 *Sizing the bins according to T_{LB}, the greedy algorithm yields a valid fractional assignment. Moreover, T_{LB} is the smallest value for which this is true. In the greedy assignment, each bin contains some number of full jobs plus at most two partial jobs.*

Two natural 2-approximation algorithms now suggest themselves, but we show

that they are not monotone. Here is the first one. Starting with the greedy assignment, round each split job to the faster of its two machines. The load on each machine is now at most the total size of the jobs fully assigned to that bin in the fractional assignment, plus at most one more job. Since the fractional assignment is valid, the rounded one overflows each bin by at most a factor of 2, so this algorithm is a 2-approximation.

Unfortunately, this algorithm does not yield decreasing load curves, as we argue now. Suppose that $b_i p_j$ is the lone bottleneck term in (4.3) with $1 < i < m$ and $j \leq n-2$. Thus, bin i has size p_j and gets part but not all of job j, the first part of this job having gone in bin $i-1$. Suppose that job $j+1$ exactly finishes off bin i. Thus, after the rounding, i gets just job $j+1$. If i perturbs its speed downwards a tiny bit, then T_{LB} increases a tiny bit, such that bin i stays size p_j, since $b_i p_j$ is still the bottleneck term defining T_{LB}. This means that all of the bins $< i$ got slightly larger, so bin i gets less of job j, but still all of job $j+1$ and now part of job $j+2$. Thus, after the rounding, i gets jobs $j+1$ and $j+2$. This situation occurs, for instance, when the jobs are sizes $5, 4, 2$ and 1, the machines are speeds $7, 4$ and 1, and the second machine lowers its speed just below 4.

Another simple 2-approximation is to round each job to a machine that processes at least half of it, since each job is split across at most two machines. Unfortunately, this algorithm also does not yield decreasing load curves. Consider the same situation as above, except that job $j+1$ is split almost evenly between bins i and $i+1$, with just over half in bin $i+1$, whereas job j is unevenly split. If i slightly decreases her speed, then bin i gets slightly more than half of job $j+1$, so it gets this job after the rounding, while job j is rounded the same way in both cases. This situation occurs, for instance, when there are jobs of sizes $5, 4$ and 2

and machines of speeds $6, 4+\epsilon$ and 2, and the second machine slows down to speed $4-\epsilon$.

It seems difficult to overcome this type of problem with a deterministic algorithm, so we turn to randomized ones. We use randomization to obtain a monotone load curve.

Our algorithm. Starting with our greedy fractional assignment of jobs to bins, we randomly assign jobs as follows. Select a single random α uniformly in $[0, 1]$, which we will use to round all jobs in a coordinated fashion. In the greedy fractional assignment, note that each job j is split between at most 2 bins. Let i be the first bin that receives part of job j, with the rest (if any) of job j spilling over into bin $i+1$ (the next smaller bin). If the fraction of job j that is assigned to i is at least α, then round job j to bin i. Otherwise, round it to bin $i+1$.

Theorem 4.2.4 *The randomized allocation described above admits a payment scheme that makes it truthful in expectation. The resulting mechanism satisfies the voluntary participation condition, and deterministically yields a polynomial-time* 2*-approximation mechanism for* $Q||C_{\max}$*. Moreover, the payments can be chosen such that truthful agents always attain a fixed utility.*

Proof: To establish the approximation guarantee, we just need to show that no bin becomes more than doubly full after the rounding. Consider bin i, with bin size B. In the fractional assignment, bin i has some number (possibly zero) of jobs that are fully assigned to that bin, plus at most two jobs that are partially assigned to i. Call these jobs j and k (if they exist). Job j is the job that is split between machine i and the next faster machine. Job k is the job that is split between machine i and the next slower machine. Bin i overflows only if one or

both of j and k is rounded to it. Since the fractional assignment is valid, p_j and p_k are both at most the size of bin i. Thus, if only one of them is rounded to it, then the overflow is at most a factor of 2. So we need worry only about the case where both j and k are rounded to bin i. If job k is rounded to bin i, then the fractional assignment must have assigned at least an α fraction of job k to bin i, by the rounding rule. Thus, it assigned at most a $1-\alpha$ fraction to the next faster machine. Similarly, the fractional assignment must have assigned less than an α fraction of job j to the next slower machine. Thus, the amount of extra work that i receives in addition to what it got in the fractional assignment is at most $\alpha p_j + (1-\alpha)p_k \leq B$. Thus, no matter what the value of α, bin i becomes at most doubly full after the rounding.

We now show that the expected load on each machine i decreases as i bids higher (i.e., claims to be slower), so that we can apply Theorem 2.4.4. Note that each job j is assigned to each machine i with probability equal to the fraction of job j that was assigned to i in the fractional assignment. Therefore, the expected load on i is precisely the load in the greedy fractional assignment. For full bins this is T_{LB}/b_i, for the empty bins it is 0, and for the (at most) one partially full bin it is the work left over from the full ones. Suppose some machine claims she is slower, replacing her bid b_i with αb_i, where $\alpha > 1$. This yields a new lower bound T'_{LB} from (4.3). Clearly $T'_{LB} \geq T_{LB}$, but also $T'_{LB} \leq \alpha T_{LB}$, since shrinking bin i by a factor of α then blowing up *all* bins by α would allow for a valid fractional assignment. Thus, the overall effect of increasing i's bid is to enlarge the other bins while shrinking bin i (or possibly leaving it the same size). If i was initially full, then its load clearly cannot increase, since the size of the bin did not increase. If i was initially the one bin that was only partially full, then the amount of work

it gets is exactly equal to the amount that did not fit into the larger bins. Since $T'_{LB} \geq T_{LB}$, the bins larger than i fit at least as much work as they did initially, so the amount left over for i cannot increase. The expected load $q_i(b_{-i}, b_i)$ is a decreasing function of b_i, so by Theorem 2.4.3, we can design a truthful payment scheme. Since machines bidding sufficiently high receive no jobs, we can choose the payments to satisfy voluntary participation, by Theorem 2.4.4.

For this mechanism, we can compute the payments exactly without resorting to the tricks of Section 2.6. To compute the payments, we must compute the function $q_i(b_{-i}, \cdot)$ and the integral $\int_{b_i}^{\infty} q_i(b_{-i}, x)dx$. Let $T_{LB}(x)$ denote our lower bound when agent i bids x and the others bid b_{-i}. For small bids (fast speeds) bin i is full, so $q_i(b_{-i}, x) = T_{LB}(x)/x$. For large bids the load is zero. For the interval inbetween, the load is just the leftover work from the larger bins. Thus, we just need to find $T_{LB}(x)$. On different intervals it is either constant, of the form cx, or of the form $\frac{c}{d+1/x}$ (where c and d are constants), depending on which term is the bottleneck in formula (4.3). Breakpoints occur only when x coincides with another agent's bid or when two of the terms inside the braces in (4.3) (considered as a function of x) cross. Thus the number of intervals is polynomial and the integral over each interval is a closed-form expression, so the mechanism is computable in polynomial-time.[2] Since we can compute $q_i(b_{-i}, b_i)$, the expected load on agent i, we can pay i a correction term of $b_i(Z - q_i(b_{-i}, b_i))$ where Z is the actual load assigned to i as a result of the random coin flips. Thus, as we commented in Section 2.6, an agent who tells the truth always attains a fixed utility, independent of the coin flips. ■

[2]The closed form expressions giving the payments in the mechanism described above may contain natural logarithms, so our model of computation must allow us to compute these if we wish to obtain a numerical answer for the payment.

Most randomized algorithms use the randomization either to improve the performance guarantee of the algorithm or the running time. Our mechanism has the peculiar feature that we introduced the randomness not for either of these reasons, but to cause the expected load to decrease monotonically as the bid increases.

4.2.1 Frugality of our load balancing mechanism

In contrast to our negative results of Chapter 3 for path selection mechanisms, the expected payments by our 2-approximation algorithm for $Q||C_{\max}$ never exceed the expected costs incurred by the machines by more than a logarithmic factor, provided no single machine dominates the processing power.

Theorem 4.2.5 *The expected payment to each machine $i \geq 2$ is at most $T_{LB}(1 + 2\ln\frac{z}{b_i})$, where $z = b_1(p_1 + \ldots + p_n)/p_n$ and T_{LB} is given by (4.3). The expected payment to machine 1 is at most $T_{LB}(\frac{b_2}{b_1} + 2\frac{b_2}{b_1}\ln\frac{z_1}{b_2})$, where $z_1 = b_2(p_1 + \ldots + p_n)/p_n$.*

Proof: The expected payment to machine i consists of two terms (formula (2.4)). The first term $b_i q_i(b_i)$ exactly compensates machine i for its expected cost, which is T_{LB} for all machines that are full in the greedy fractional assignment of Lemma 4.2.3. The second term, $\int_{b_i}^{\infty} q_i(x)dx$, is the (expected) profit. We always have $q_i(x) \leq T_{LB}(x)/x$, where $T_{LB}(x)$ is the lower bound on the optimal makespan $C^*_{\max}$ computed in formula (4.3) when machine i bids x and the other machines bid b_{-i}. Equality holds as long as bin i is full in the fractional assignment. But (for $i > 1$) $T_{LB}(x)$ stays approximately constant, since machine i constitutes at most half of the processing power, so decreasing its speed can at most double $T_{LB}(x)$. To see this, recall from Lemma 4.2.3 that T_{LB} is the smallest T such that sizing the bins according to T admits a valid fractional assignment. Consider the valid fractional assignment obtained from $T_{LB}(b_i)$. Now double all the bin sizes, delete bin i, and

reassign all the fractional jobs it had to the largest bin. Since the free space in the largest bin is at least as large as the original size of bin i, all of these jobs fit, and this fractional assignment is valid. Thus, $T_{LB}(\infty) \leq 2T_{LB}$. Moreover, $q_i(x)$ drops to zero at some point y. Thus, the integral is at most $2T_{LB} \ln \frac{y}{b_i}$.

This argument breaks for machine 1 if it is much faster than all the rest combined, since the load on this machine stays nearly constant as its announced speed decreases toward that of the second fastest machine. Therefore, our bound on its profit depends on the ratio between the speeds of the fastest two machines.

It then remains to bound y. If machine $i > 1$ has speed $1/x = p_n/(b_1(p_1 + \ldots + p_n))$, then placing even the smallest job on i would take longer than processing all jobs on machine 1. Thus, when i bids z, it gets no work, so $y \leq z$. Similarly, when machine 1 bids z_1, it gets no work. ■

Corollary 4.2.6 *If the sizes of all n jobs differ by a factor of at most r_1, and the speeds of the two fastest machines differ by a factor of r_2, then the payment given by the mechanism exceeds the total expected cost incurred by all the agents by a factor of at most $O(r_2 \ln(r_1 n))$.*

Proof: We apply Theorem 4.2.5 to bound the ratio of payment to expected cost, machine by machine. For each machine i whose bin is full in the greedy fractional assignment, the cost is T_{LB}. There is at most one machine with a partially full bin. We cannot bound its ratio, but clearly its payment is no greater than that of the next faster machine. For machine 1, the ratio is at most $r_2(1 + 2\ln(r_1 n))$, and for each other full machine the ratio is at most $1 + \ln(\frac{r_1 n}{r_2})$. ■

4.3 Load balancing with preemption

Now we consider another load balancing variant, often denoted $Q|\text{pmtn}|C_{\max}$. Just as in $Q||C_{\max}$, the machines run at different speeds and we are aiming to minimize the makespan. However, now we allow *preemption*, which means that a job can be started on one machine, interrupted and continued later on the same machine or another. The output consists of a complete schedule, defining which jobs go on which machines for which periods of time. The only constraints are that no machine may work on more than one job at a time, no job may be run on two different machines at once, and the total work done on each job j must meet its processing requirement p_j. There are multiple algorithms to solve this problem optimally in polynomial time [29, 37], but not all are monotone as stated. Here we give a method for converting any optimal algorithm for $Q|\text{pmtn}|C_{\max}$ into a monotone one.

We begin by examining a combinatorial lower bound on the optimal makespan T. In describing the algorithm, we assume the job sizes and machine speeds are sorted largest to smallest. Since each job may be processed on at most one machine at a time, the fastest k machines must have enough collective capacity to handle the longest k jobs within time T. Thus,

$$T_k = \frac{\sum_{j=1}^{k} p_j}{\sum_{i=1}^{k} s_i}$$

is a lower bound on T, for each $k = 1, \ldots, m$. Moreover, all the m machines must have enough collective capacity to handle all n jobs, which yields another lower bound

$$T_0 = \frac{\sum_{j=1}^{n} p_j}{\sum_{i=1}^{m} s_i}.$$

The maximum of these lower bound turns out to be tight, since it is obtainable by

a feasible schedule.

Theorem 4.3.1 ([29, 37]) *There is an algorithm for $Q|\text{pmtn}|C_{\max}$ that runs in polynomial time and outputs an optimal schedule with makespan equal to*

$$\max_{0 \le k \le m} T_k.$$

We now show how to transform any optimal algorithm $\mathcal{A}$ into a monotone one. Suppose that T_0 is the bottleneck. Then all of the machines are busy for the entire time T_0. In this case, we simply use the schedule output by $\mathcal{A}$. Otherwise, T_k is the bottleneck for some k (in the case of a tie, choose the largest k). Then algorithm $\mathcal{A}$ must keep all k of the fastest machines busy for the entire time T_k, working only on the k longest jobs. In this case, we schedule these k jobs on these k machines as given by $\mathcal{A}$, and recursively schedule the remaining jobs on the remaining machines. Each level of the recursion results in scheduling a new block of machines. The first block consists of the f_1 fastest, the second block of the f_2 fastest, and so on. The machines in the k^{th} group all process for an amount of time $T^{(k)}$, which is the lower bound that was computed during that level of the recursion.

Proposition 4.3.2 *If there is more than one block, the block processing times satisfy $T^{(1)} > T^{(2)} > \ldots$.*

Proof: We prove that $T^{(1)} > T^{(2)}$. The rest of the inequalities follow by induction. Since $T^{(1)}$ is the bottleneck at the top level of recursion, this shows that the longest f_1 jobs barely fit on the first f_1 machines. Thus, algorithm $\mathcal{A}$ must schedule them that way. Moreover, it schedules the rest of the jobs within the same time limit. But $T^{(2)}$ is the optimal makespan for the residual problem, hence $T^{(2)} \le T^{(1)}$.

Strict inequality holds because block 1 was chosen according to the largest k such that T_k was the bottleneck. ■

Theorem 4.3.3 *The allocation described above is optimal, monotone, and admits a truthful payment scheme satisfying the voluntary participation condition. The mechanism can be computed in polynomial time.*

Proof: Clearly the allocation can be computed in polynomial time, since it involves at most m calls to the polynomial-time algorithm $\mathcal{A}$. (Actually, $\mathcal{A}$ is called for disjoint blocks of the input, so the overall time is bounded by the time for running $\mathcal{A}$ on the whole input.) It computes an optimal allocation, because it has makespan $T^{(1)}$, just like algorithm $\mathcal{A}$. We just need to show that the load curves are decreasing, and the payments can be computed efficiently.

Recall that the speed used in the algorithm is the inverse of a machine's bid. Thus, as machine i increases its bid, it decreases its speed. We must show that this decreases its load. Suppose that, for a particular bid, machine i is part of the k^{th} block. Let $T^{(k)}(x)$ denote the value of $T^{(k)}$ computed when i's bid is x. As i slows down (i.e., x increases), it increases $T^{(k)}(x)$ until either $T^{(k)}(x) = T^{(k-1)}$ (and the k^{th} block merges with the $(k-1)^{st}$), or until machine i becomes as slow as the fastest machine in block $k+1$, and they switch places. Between these breakpoint values, as b_i changes, only $T^{(k)}$ changes with it – none of the other $T^{(\ell)}$'s do. At a breakpoint where two blocks merge, i gets the same load. At a breakpoint where i moves to the next block, i's load only decreases, by Proposition 4.3.2. Between breakpoints, i's processing time $T^{(k)}$ is of the form $\frac{a}{c+\frac{1}{b_i}}$ where $a > 0, c \geq 0$. Thus, its load is of the form $\frac{a}{1+cb_i}$, which is decreasing in b_i, as required.

For each machine i, we need to integrate the load curve over x. The integral over each segment between breakpoints can be computed in closed form, since

the load is of the form $\frac{a}{1+cb_i}$. Since, at each breakpoint, either two blocks merge or the machine jumps from one block to the succeeding one, the total number of breakpoints is at most twice the number of blocks, which in turn is at most m. Thus, the payments can be computed in polynomial time. ■

4.4 Sum of completion times

We now turn to another problem involving uniformly uniformly related parallel machines, scheduling to minimize the sum of completion times. This problem is sometimes denoted $Q||\sum C_j$. The setup is exactly the same as the one we considered for $Q||C_{\max}$, except we consider a different objective function g. Now, in addition to allocating jobs to machines, we must specify in which order each machine processes it's jobs. The time at which job j completes is denoted by C_j, and the objective is to minimize $\sum_j C_j$. This optimization problem is solved by a simple algorithm [15], which we describe below. The goal of this section is to prove that this algorithm yields a truthful mechanism that can be computed in polynomial time.

Assume we have numbered the jobs so that $p_1 \geq \ldots \geq p_n$. Consider a given schedule of jobs on machines, and let us rewrite the objective function in a more convenient form. In this schedule suppose that job j is processed on machine i, and let $P(j)$ denote the set of jobs that are processed on machine i prior to job j. Then the completion time C_j of job j is $\sum_{k\in P(j)\cup\{j\}} \frac{p_k}{s_i} = \sum_{k\in P(j)\cup\{j\}} b_i p_k$, since

$b_i = \frac{1}{s_i}$. If $J(i)$ denotes the set of all jobs processed on machine i, then we have

$$\begin{aligned}\sum_{j \in J(i)} C_j &= \sum_{j \in J(i)} \sum_{\ell \in P(j) \cup \{j\}} b_i p_\ell \\ &= \sum_{\ell \in J(i)} \sum_{j \in J(i) - P(\ell)} b_i p_\ell \\ &= \sum_{\ell \in J(i)} |C(i) - P(\ell)| b_i p_\ell.\end{aligned}$$

In other words, if job j is the k^{th} to last job scheduled on machine i, then it contributes $kb_i p_j$ to the objective function, because it takes time $b_i p_j$ to process, and this time contributes to the completion times of k jobs.

Therefore, we can think of the problem as assigning jobs to slots (at most one job per slot), where there are slots for the k^{th} to last job on each machine i, $i = 1, \ldots, m$, $k = 1, \ldots, n$. Slot k on machine i has a multiplier kb_i, and the contribution of job j to the objective function is p_j times the multiplier for the slot it uses. Define the multiset of multipliers $M = \{kb_i : k = 1, \ldots, n, i = 1, \ldots, m\}$. Clearly, the best assignment of jobs to slots is the greedy one: the j^{th} largest job goes in the slot with the j^{th} lowest multiplier in this multiset.

Theorem 4.4.1 *The algorithm described above is monotone, admits a truthful payment scheme satisfying the voluntary participation constraints, and yields an optimal solution for $Q||\sum C_j$. The mechanism is computable in polynomial time.*

Proof: We have already argued that the solution is optimal.

We must show that the load on machine i only decreases as machine i raises her bid and eventually reaches zero, so that we can use Theorem 2.4.4 to get a truthful payment scheme. We compute the load on machine i as a function of her bid by considering $M(x)$, the multiset of multipliers when i bids x and the other machines bid b_{-i}. The assignment changes at breakpoints x where one of the

multipliers corresponding to machine i coincides with a multiplier corresponding to another machine, at which point the jobs in the two slots are exchanged. Since x is increasing, the multipliers for i's slots are increasing, so in this exchange, i gives away a job in exchange for the next smaller job. Therefore, the load function is a step function that drops at these breakpoints and is constant elsewhere. The load eventually drops to zero when i's smallest multiplier x becomes the $(n+1)^{st}$ smallest multiplier in $M(x)$. Thus, by Theorem 2.4.4, we can create a truthful payment scheme satisfying voluntary participation constraints. The output algorithm is clearly computable in polynomial time. Finding the payments requires computing integrals of the load curves, as in (2.5). Since each load curve has at most $O(n^2)$ breakpoints, the payments can be computed in polynomial time. ■

4.5 Maximum Flow

Now we consider the maximum flow problem. We are given a directed graph $G = (V, E)$ with m edges, a source node S, and a sink node T. Each edge e has a finite non-zero capacity c_e. The problem is to find a max flow from S to T respecting the capacity constraints on all edges. There are many algorithms for computing a max flow in polynomial time (see, e.g., [2]). In this section, we cast max flow as a mechanism design problem, and show how to obtain polynomial time truthful mechanisms.

We assume that each edge e is an agent, and the capacity of her edge is private data. Agent e incurs a cost equal to the congestion on her edge. That is, if we send q_e units of flow on edge e, then agent e incurs a cost of q_e/c_e. In order for this problem to fit the form we have been considering (i.e., the agent's cost equals $t_e q_e$, her type times her load), we take the private data to be $t_e = 1/c_e$, and the load on

edge e to be the flow q_e. Since the edge capacity is the natural quantity to think about, we will think of an edge's bid as announcing its capacity. But for purposes of using Theorem 2.4.3 and equation (2.5) to construct truthful mechanisms, we take b_e to be the reciprocal of the capacity that edge e announced. Thus, in truthful mechanisms the flow on edge e must decrease as its announced capacity decreases.

We can guarantee this property by using max flows that are lexicographically minimal (in the sense of Definition 4.2.1). We can compute such a flow using m max flows (where m is the number of edges). There is a closed-form expression for the payments in terms of the flow, so we can design a mechanism that solves max flow exactly in polynomial time. Unfortunately, the load curves have infinite integrals, so we cannot satisfy the voluntary participation condition. However, we could choose to ignore edges of capacity below ϵ/m, in which case the load curves would drop to zero at that point, so we could satisfy voluntary participation and obtain a flow within an additive ϵ of optimal.

Proposition 4.5.1 *No truthful mechanism for max flow that always yields the true max flow can satisfy the voluntary participation constraints.*

Proof: Let e be any edge in the min cut when the bids are $(b_{-e}, \hat{b}_e)$. Then if e announces a capacity below $1/\hat{b}_e$ (bids above $\hat{b}_e$), it is still part of the min cut, so the algorithm must saturate it. But then $q_e(b_{-e}, b_e) = 1/b_e$ for $b_e \geq \hat{b}_e$, so $\int_0^\infty q_e(b_{-e}, x)dx = \infty$, and by Theorem 2.4.4, there is no truthful payment scheme that satisfies the voluntary participation constraints. ■

It is not enough to compute any max flow according to the announced capacities $1/b_e$ and use this as our output, since this will not necessarily lead to the flow on edge e decreasing as its bid increases (i.e., as its announced capacity decreases). Instead, we number the edges $1, \ldots, m$ and compute a lexicographically minimal

max flow. That is, among all max flows, we choose the one that sends the minimum flow across e_1, and among those we choose the one that sends the minimum flow across e_2, etc.

Proposition 4.5.2 *We can compute a lexicographically minimal max flow via m max flow computations.*

Proof: We first compute any max flow q from S to T. We then shunt as much flow as possible off of edge $e_1 = (u_1, v_1)$ by sending it from u_1 to v_1 through the rest of the network. Specifically, we remove edge e_1 from the residual network and update q by superimposing a max flow from u_1 to v_1 in the residual network, and decrementing q_{e_1} by this flow value. This yields a max flow from S to T in which q_{e_1} is minimized. We proceed similarly with edges $e_2, \ldots, e_{m-1}$. When shunting flow off of edge $e_i = (u_i, v_i)$, we need to make sure that we do not change the flows already computed for edges $e_1, \ldots, e_{i-1}$. Thus, we delete edges $e_1, \ldots, e_{i-1}$ before computing the max flow from u_i to v_i in the residual graph. ■

We now reason about the shape of the load curves when we use lexicographically minimal max flows. Fix an edge e, and the other bids b_{-e}. Suppose that edge e is not saturated in the lexicographically minimal max flow. We claim that if e increases its announced capacity, then the lexicographically minimal max flow does not change. If it did, then every strict convex combination of the new flow and the old flow would improve over the old flow, and those sufficiently close to the old flow would still obey the old capacity on edge e, so would be feasible. This contradicts the old flow being lexicographically optimal for the old bid. Thus, there is some constant cutoff K such that the flow sent on e is K if e's announced capacity is at least K (i.e., $b_e \leq 1/K$) and the flow saturates the edge if the announced capacity is at most K ($b_e \geq 1/K$).

We just showed the flow (load) on edge e decreases as its announced capacity decreases (bid increases). By Theorem 2.4.3, it admits a truthful payment scheme, but by Proposition 4.5.1, we cannot choose it to satisfy voluntary participation constraints since it computes the true max flow. However, if we are prepared to lose an additive ϵ in the flow, then we can satisfy the voluntary participation constraints. Let F^* denote the value of the max flow with respect to the announced capacities $1/b_e$.

Theorem 4.5.3 *Given any fixed ϵ, there is a truthful polynomial time mechanism for max flow that finds a flow of value at least $F^* - \epsilon$ and satisfies the voluntary participation constraints. Moreover, the output and payments may all be computed with a single lexicographically minimal max flow.*

Proof: We immediately discard edges whose announced capacity is less than ϵ/m (i.e., edges e with $b_e > m/\epsilon$). The total capacity of all discarded edges is at most ϵ, so the value of the minimum cut decreased by at most ϵ, so the max flow in the remaining network is at least $F^* - \epsilon$. Now compute a lexicographically minimal max flow in the remaining network. As in the discussion above, the flow on any edge e is some constant K_e for large announced capacities ($b_e \leq 1/K_e$), then saturates edge e for announced capacities between K_e and ϵ/m (flow $1/b_e$ for $1/K_e \leq b_e \leq m/\epsilon$), but then it drops to zero (for $b_e > m/\epsilon$). Thus, using (2.4) for our payment scheme, we get a truthful mechanism satisfying voluntary participation constraints. Moreover, we may compute the payment to edge e directly from the flow q_e on e, using (2.4). Edges with no flow are paid zero. If e is saturated (i.e., $b_e q_e = 1$), then $P_e = 1 + q_e \ln \frac{mq_e}{\epsilon}$. If e is not saturated and carries a non-zero flow less than ϵ/m, then $P_e = q_e m/\epsilon$. Otherwise, $P_e = 1 + q_e \ln \frac{mq_e}{\epsilon}$. ■

4.6 Affine Functions of the Loads: LP and Uncapacitated Facility Location

Here we consider a general class of problems admitting truthful mechanisms. Our first result is the existence of a truthful mechanism; at our initial level of generality we cannot say whether we can compute it. Later we discuss special cases where we can compute it in polynomial time. Suppose the mechanism wishes to minimize some affine function of the loads

$$g(o, b) = d(o) + \sum_{i \in \mathcal{N}} c_i(b_i) q_i(b), \tag{4.4}$$

where d is some function of the output o not depending on the bids, $c_i(b_i)$ is an increasing function for each agent i, and the set of feasible outputs o does not depend on the bids. Assume that, for every set of bids, there exists an optimal solution to (4.4). Recalling Definition 4.2.1, let us number the agents from 1 to m.

Theorem 4.6.1 *For the problem stated above, if each coefficient $c_i(b_i)$ is strictly increasing in b_i, then any optimal output function $O(b)$ admits a truthful payment scheme. Otherwise, any output function that gives an optimal solution whose vector of loads $(q_1, \ldots, q_m)$ is lexicographically minimal admits a truthful payment scheme.*

Proof: To apply Theorem 2.4.3, we just need to show that the load $q_i(b)$ on agent i never increases as her bid b_i increases, with the other bids b_{-i} fixed. Let $\bar{b} = (b_{-i}, \bar{b}_i)$ and $\hat{b} = (b_{-i}, \hat{b}_i)$ be two sets of bids differing only in agent i's bid, and let $\bar{o}, \hat{o}, \bar{q}_j$ and $\hat{q}_j$ be the corresponding outputs and loads. Assume on the contrary that $\hat{b}_i > \bar{b}_i$ but $\hat{q}_i > \bar{q}_i$. Let $\Delta = (d(\bar{o}) + \sum_{j \neq i} c_j(b_j)\bar{q}_j) - (d(\hat{o}) + \sum_{j \neq i} c_j(b_j)\hat{q}_j)$, the difference in the two objective functions, not including the q_i term, since that is the only one that depends on b_i. Since $\hat{o}$ is a feasible output for bid $\bar{b}$ but the function chose $\bar{o}$

instead, we must have $0 \leq g(\hat{o}, \bar{b}) - g(\bar{o}, \bar{b}) = -\Delta + c_i(\bar{b}_i)(\hat{q}_i - \bar{q}_i)$. Similarly, since the output $\hat{o}$ is optimal for bids $\hat{b}$, we have $0 \leq g(\bar{o}, \hat{b}) - g(\hat{o}, \hat{b}) = \Delta + c_i(\hat{b}_i)(\bar{q}_i - \hat{q}_i)$. Combining these gives $(c_i(\bar{b}_i) - c_i(\hat{b}_i))(\hat{q}_i - \bar{q}_i) \geq 0$. If c_i strictly increased as b_i increased from $\bar{b}_i$ to $\hat{b}_i$, then the product is negative, which is a contradiction. If c_i stayed the same, then the product is zero, so all of the inequalities are tight. Thus, both outputs $\bar{o}$ and $\hat{o}$ give the same objective function value for both bids $\bar{b}$ and $\hat{b}$ but the loads are different, which contradicts their both being lexicographically minimal. ■

There are many examples of problems fitting this model. One example is linear programming problems in which some of the decision variables are the loads q_i, $d(o)$ is the part of the objective depending on the other variables, the cost coefficient on q_i depends on i's bid b_i, and the feasible set is given by linear inequalities *not* depending on the bids. One example would be minimum cost flow, where each agent represents an edge e, her cost is $t_e q_e$ where e is the flow on edge e, and the mechanism is aiming to minimize $\sum_e c_e(b_e)q_e$, subject to the standard flow constraints, with capacities on the edges and supplies and demands of flow at the nodes. It is important that the capacities, supplies and demands do not depend on the bids, since these determine the feasible set of flows.

Another example is uncapacitated facility location. In this problem, there is a set of facility points $\mathcal{F}$, a set of client points $\mathcal{C}$, costs f_i for each facility i, and a metric d giving the distance between every pair of points in $\mathcal{C} \cup \mathcal{F}$. A solution selects a collection of facilities to open. Then each client is assigned to the closest open facility. There are no capacities, so an open facility can serve an arbitrary number of customers. We incur the facility cost f_i for each open facility i, and if client j is assigned to facility i, we incur a service cost $d(i, j)$. The objective is to

minimize the sum of the demand and service costs. In our model, each facility is an agent, her private data is her facility cost, and she will incur this cost if she is opened. Thus in (4.4), q_i is an indicator variable telling whether i is opened, $c_i(b_i) = b_i$ (the reported facility cost), and $d(o)$ is the service cost.

If the functions $c_i(b_i)$ are given by $w_i b_i$ for some set of weights w_i, then our objective function is the same one optimized by the weighted, biased VCG mechanism. However, our mechanism treats a more general case, since it works for all increasing functions c_i.

Even though the maximum flow problem (Section 4.5) and load balancing with preemption on uniformly related machines to minimize makespan (Section 4.3) can both be solved by linear programming, they do not fit into this model because the feasible regions depend on the bids. For max flow, there is a constraint that the flow variable on edge e is at most e's capacity, which in our model is given by e's bid. For the linear programming formulation of the load balancing problem, the objective function is a variable z, and the program includes constraints for every machine i saying that z is at least as large as i's total processing time. Since the expression for the processing time involves i's speed, which is given by her bid, the feasible region depends on the bids.

We mention linear programming because it can be used to model many optimization problems, and because linear programs can be solved in polynomial time. However, there is one catch with computing the payments. While we can optimize over the polyhedron of feasible solutions in polynomial time, we cannot necessarily compute the payments in polynomial time. Computing the payment to agent i requires integrating the work curve, so we need to know how $q_i(b_i)$ changes with b_i (suppressing the other bids b_{-i} in the notation, as they are held fixed).

As b_i increases, the objective function coefficient vector changes, but the feasible polyhedron remains the same. Thus, $q_i(b_i)$ stays constant over intervals where the optimal vertex of the polyhedron is the same. There are finitely many breakpoints where the optimum jumps to another vertex of the feasible polyhedron. So the load curve is a step function. Unfortunately, polyhedra can be exponentially complex in the size of their description, so there could be exponentially many breakpoints. Thus, it might take exponential time to compute the payments, even though we can compute the output in polynomial time. In these cases, we can resort to the tricks in Section 2.6. We can either discretize the bids in order to obtain a truthful deterministic mechanism that approximately optimizes our objective, or we can use random sampling to obtain an optimal mechanism that is truthful in expectation.

4.6.1 Uncapacitated facility location

We can apply Theorem 4.6.1 to the uncapacitated facility location problem, as defined in the last section, because $c_i(b_i) = b_i$, which is strictly increasing in b_i. This yields the following.

Theorem 4.6.2 *Any algorithm that solves the uncapacitated facility location problem optimally admits a truthful payment scheme.*

Uncapacitated facility location is NP-hard, so we cannot expect to find a polynomial-time algorithm to solve it. However, there are some special cases that can be solved in polynomial time, such as if the facilities and customers lie on a line, circle, or tree. Slight generalizations of these cases are solved in [27]. In these cases, we can also compute each payment easily, as it just involves finding the threshold bid at which a facility would no longer be open (i.e., where $q_i(b_{-i}, \cdot)$ jumps from 1 to 0).

We can adapt the algorithm of Meyerson [48] to create a truthful mechanism for facility location in an arbitrary metric space. Meyerson's algorithm works for the case where each customer point is also a potential facility, and all facilities cost a uniform amount f. The algorithm is simple. We select a random permutation of the facilities, and consider them one by one. When a new facility arrives, we compute the cost c of connecting the associated customer to the nearest facility already open. We then open the new facility with probability $\min(1, \frac{c}{f})$. Meyerson proves that this randomized algorithm gives a constant approximation factor (in expectation).

In our setting, the facility costs are reported by the agents. Suppose that we know ahead of time that all facility costs lie in a bounded interval $[f_1, f_2]$, where $\frac{f_2}{f_1}$ is a constant. In this case, for a facility reporting its cost to be b_i, we open it with probability $\min(1, \frac{c}{b_i})$. Meyerson's analysis is valid only for uniform facility costs, but since $\frac{f_2}{f_1}$ is a constant, we only lose an additional constant factor.

Here the expected load on an agent (facility) is the probability that it is opened, which is clearly monotone decreasing. In this case, we need only integrate up to f_2 in the formula (2.5), since agents cannot bid higher than that. It is not clear how to compute the overall expected probability of opening i, as a function of the bid b_i. Fortunately, there is an even simpler method of calculating payments in this case. We simply compute the largest value (in $[f_1, f_2]$) that i could have bid and still been opened, conditioned on all of the coin flips up to that point in the randomized algorithm. Thus, this method is truthful for every outcome of the coin flips, not just truthful in expectation.

Theorem 4.6.3 *There is a randomized universally truthful mechanism for uncapacitated facility location that achieves a constant approximation factor in expecta-*

tion, provided every customer point is also a potential facility, and all the facility costs are known to lie in an interval $[f_1, f_2]$*, where* $\frac{f_2}{f_1}$ *is bounded by a constant.*

4.7 Lower bounds

We can use the characterization of Theorem 2.4.3 to prove lower bounds on approximability by truthful mechanisms. In particular we consider scheduling on machines with speeds to minimize the weighted sum of completion times, usually denoted $Q||\sum w_j C_j$. This is identical to the problem studied in Section 4.4, but with a slightly different objective function. Each job j has a weight $w_j > 0$, and the goal is to schedule jobs on machines to minimize $\sum_j w_j C_j$. Whereas minimizing $\sum_j C_j$ was easy, minimizing $\sum_j w_j C_j$ is NP-hard. However, there is a PTAS for this problem [13]. We show a lower bound of $\frac{2}{\sqrt{3}} \approx 1.15$ for truthful mechanisms even scheduling just two jobs on just two machines. This is interesting because it shows that for this problem, truthfulness implies a stronger lower bound on approximability than does the restriction to polynomial-time algorithms.

The idea behind our lower bound is that in an optimal allocation, the fast machines should get the important jobs, whereas in a truthful allocation the fast machines should get the bulk of the work. If we arrange the job weights w_j so that these two principles conflict, then the truthful allocation will be suboptimal.

Theorem 4.7.1 *No truthful mechanism for* $Q||\sum w_j C_j$ *can achieve an approximation ratio better than* $\frac{2}{\sqrt{3}}$*, even on instances with just two jobs and two machines.*

Proof: Consider an instance with two jobs and two machines. Job 1 has weight and processing requirement 1, while job 2 has weight w and processing requirement p, where $p > 1$. Suppose machine 1 runs at speed 1, and machine 2 runs at speed s. In order for our mechanism to be truthful, the load on machine 2 must increase

monotonically as its bid decreases (i.e., as its speed increases). To show that any truthful mechanism is suboptimal, we must select p and w so that in the optimal schedule, the load on machine 2 is non-monotone in s. With foresight, we set p and w such that $pw < 1$.

In the optimal schedule, for small s, both jobs will be assigned to machine 1. As we raise s, the jobs will eventually be split between machines, the machines will swap jobs, then eventually machine 2 will get both jobs, for large enough s. When the jobs are split, the job with larger weight-processing product goes on the faster machine. Since $pw < 1$, job 2 goes to the slower machine. Thus, the load on machine 2 is non-monotone in the optimal assignment, since $p > 1$. It is easy to check that the optimal assignment is to give both jobs to machine 1 when $s \in (0, \frac{p}{p+1})$, put job 1 on machine 1 and job 2 on machine 2 when $s \in (\frac{p}{p+1}, 1)$, swap the jobs when $s \in (1, 1 + \frac{1}{p})$, and put both jobs on machine 2 for $s \in (1 + \frac{1}{p}, \infty)$. Whenever both jobs are on the same machine, job 1 goes first (by Smith's rule).

Now we reason about the behavior of any truthful mechanism, as s increases from 0 to ∞. If the mechanism is to achieve a finite approximation ratio, then it must assign both jobs to machine 1 when $s \ll 1$ and both jobs to machine 2 when $s \gg 1$. For intermediate values of s it may split the jobs. The key is that if machine 2 gets job 2 for some value of s, then it must keep job 2 for all larger values of s, since the load may only increase and $p > 1$. The optimal schedule violates this. Because $pw < 1$, we have $\frac{1}{1+w} \in (\frac{p}{p+1}, 1)$. Thus, when $s = \frac{1}{1+w}$, the optimal schedule gives job 2 to machine 2, but when $s = 1 + w$, it gives only job 1 to machine 2. Therefore, the truthful mechanism is suboptimal either when $s = \frac{1}{1+w}$ or when $s = 1 + w$. We discuss the first case. The second is symmetric.

Since we are assuming the truthful mechanism does not split the jobs the

optimal way when $s = \frac{1}{1+w}$, the best possible schedule it can use is to split the jobs the other way. Thus, the mechanism gives a schedule with objective function value at least $pw + (1 + w)$, while the optimal schedule has value $1 + pw(1 + w)$. For any fixed p, the ratio of these two values is maximized when $w = \frac{-p+\sqrt{2p^2+p}}{p^2+p}$. The ratio increases as $p \downarrow 1$, approaching a limit of $\frac{2}{\sqrt{3}}$. ■

4.8 Further work

This chapter addressed the interesting problem of "discrete optimization with a blindfold," trying to solve discrete optimization problems where we do not know all of the data up front, and instead must elicit it from some rational agents. Admittedly, our models are somewhat artificial because we limited ourselves to agents with one-dimensional types, since we wanted to use Theorem 2.4.3. It would be nice to be able to address more realistic models. Unfortunately, there is a dearth of useful characterization theorems analogous to Theorem 2.4.3 for more complicated valuation functions, if our goal is to optimize non-utilitarian objective functions.

Our favorite example of a considerably more compelling model for which there is still some hope of proving something is the problem of load balancing on unrelated machines ($R||C_{\max}$), introduced in the mechanism design context by Nisan and Ronen [58] and described in Section 4.1. Any time there is such a large gap between what we know how to do and what we know is impossible, it is tantalizing to try to close this gap. In this case, Nisan and Ronen gave a truthful mechanism obtaining an m-approximation, but their lower bound for the approximation ratio achievable by a truthful mechanism is only 2. Using strong additional assumptions on the form of the payment scheme used, they can improve the lower bound to m.

However, these assumptions explicitly rule out the types of payments that would encourage load balancing, so we do not see this as convincing evidence that the lower bound for all truthful mechanisms should be m.

There is a characterization of truthful mechanisms akin to Theorem 2.4.3 that almost applies to this model, so we describe it here in order to point out where the difficulties lie. The type for machine i in this model is itself a vector, $t_i = (t_{i1}, \ldots, t_{in})$, where t_{ij} is the time required for machine i to process job j. Recall that machine i's cost is assumed to be the total amount of time it spends processing. Thus, if q_{ij} is an indicator function denoting whether job j is assigned to machine i, then we can write i's valuation as $v_i = -\sum_{j=1}^{n} t_{ij} q_{ij}$.

We now describe a theorem from [47] that applies to valuation functions of this form. Consider the real-valued function $q_{ij}(b_{-i}, b_i)$, with b_{-i} fixed. Since the types are n-dimensional, b_i is an n-vector, the value that i reports for the vector $t_i = (t_{i1}, \ldots, t_{in})$. Let q_i be the vector of functions $(q_{ij})_{j=1}^{n}$. Suppose that each of the vector-valued functions $q_i(b_{-i}, b_i)$ is continuously differentiable with respect to the vector b_i (for each fixed b_{-i}). Then the derivative of q_i with respect to b_i (with b_{-i} fixed) is an $n \times n$ matrix $D_i(b_{-i}, b_i)$. The theorem from [47] says that the allocation function q is implementable if and only if $-D_i(b_{-i}, b_i)$ is positive semidefinite, for all b_{-i} and b_i.

There are two problems with this theorem in light of our intended application. First, it seems rather unwieldy to apply. Second, it does not apply directly to the model for $R||C_{\max}$ because in this model, each q_{ij} is a binary function, and the theorem assumed that the q_{ij} were continuously differentiable. It would be interesting if one could prove a version of this theorem that gave a useful characterization of implementable binary allocations, and then apply this characterization either to

create a truthful mechanism for $R||C_{\max}$ with an approximation guarantee better than m, or to improve the lower bound beyond 2.

Chapter 5

Randomized approximations to VCG

In this chapter, we consider how to obtain truthful mechanisms that approximate the VCG mechanism in the case that the VCG mechanism is NP-hard to compute. We focus on a particular case of combinatorial auctions involving so-called "single-minded bidders." Our main contribution is to show how an allocation based on randomized rounding of a linear program can be made simultaneously truthful with high probability and truthful in expectation.

Multiple-item auctions are a basic tool of electronic commerce. These auctions allow bidders to coordinate their purchases of complementary goods, such as air-fare, lodging and ground transportation for a vacation package, or mobile phone spectrum licenses in adjacent geographical regions. Many such large-scale auctions have indeed been used recently by the FCC and governmental bodies in Europe and elsewhere to allocate spectrum licenses to mobile phone providers. The FCC auctions alone granted thousands of licenses to hundreds of companies, raising over $40 billion [16]. While the FCC decided to use an innovative simultaneous ascending auction for its various spectrum licenses, it also strongly considered various forms of combinatorial auctions, in which the participants bid on sets of items, rather than individual items. The sheer magnitude of these spectrum auctions and the rise of electronic commerce have both generated a surge of interest in designing good mechanisms for such combinatorial auctions.

We will consider combinatorial auction mechanisms where each bidder i bids a valuation b_i for a set S_i she is interested in. We will assume that each bidder i is bidding for a single set S_i, and this set is known to the auctioneer or can be inferred from context. Thus, each agent's only private information is her true

valuation for that set.

There are two natural goals for designing good auctions: maximizing the revenue, and maximizing the total valuation, which is the sum of the valuations of the bidders who receive their desired sets. In this chapter we will concentrate on the latter objective, which is referred to as *efficiency* in the economics terminology. *In some cases, maximizing efficiency is a more important objective than generating revenue.* For instance, one of the primary goals in the spectrum auctions was to get spectrum licenses into the hands of the companies that could best use them to build up a viable mobile phone network, and it is widely believed that high valuation is a strong indicator of how well-positioned the company is to make good use of the spectrum license [16].

The VCG mechanism is truthful and maximizes the total valuation. Thus, in this context we would like to use it, unlike in Chapter 4, where we were interested in non-utilitarian objectives. However, computing the VCG mechanism in this case requires solving the well-known SetPacking problem, which is NP-hard. Simply replacing the exact optimization routine required in the VCG mechanism with an approximation algorithm destroys truthfulness [57].

We create a technique that makes randomized rounding-based approximation algorithms useful in designing truthful mechanisms. First introduced by Raghavan and Thompson [62], randomized rounding of an LP solution is now a commonly used technique for designing polynomial-time approximation algorithms. Typically such rounding algorithms succeed with high probability. However, it is not clear what the associated mechanism should do to ensure truthfulness when the rounding fails to produce a feasible solution. By Theorem 2.4.5, a randomized allocation algorithm $\mathcal{A}$ can be implemented truthfully in expectation if and only if for every

agent i, the probability that $\mathcal{A}$ assigns the desired set S_i to agent i is increasing in her bid b_i. Thus, our first task is to develop a technique for obtaining a monotone allocation algorithm from such a rounding scheme, which we do in Section 5.3. In Section 5.4 we construct a clever payment scheme that allows us to achieve truthfulness both in expectation and with high probability. In Section 5.5 we show that our mechanism approximates the "fractional" version of the VCG mechanism, both in terms of valuation satisfied and revenue generated.

5.1 Combinatorial auctions and single-minded bidders

A combinatorial auction is designed to divide up a set $\mathcal{G}$ of items among a set $\mathcal{N}$ of n bidders. Each bidder's valuation depends only on the set she is assigned. Thus, i's valuation is given by a function $v_i : 2^{\mathcal{G}} \to \mathbb{R}^+$. A *single-minded bidder* i is one who values only a particular set of items. More formally, there is a set S_i and a $c \geq 0$ such that $v_i(T) = c$ if $S_i \subseteq T$ and $v_i(T) = 0$ otherwise. In this chapter, we consider the case where all bidders are single-minded, and the auctioneer knows the sets S_i ahead of time. This is the case of *known single-minded bidders*. There can be multiple copies of each good, in which case the *multiplicity* m_j denotes the number of copies of item j that are available.

Two previous papers have studied auctions with single-minded bidders, in the case that all goods have multiplicity one. Lehmann et al. [41] first studied this problem, giving a truthful mechanism based on a greedy allocation. Their mechanism attains a $\sqrt{m}$-approximation to the optimal allocation, where m is the number of items. Moreover, they prove their result is nearly the best possible, in the following sense. For every constant $c < \frac{1}{2}$, there is no polynomial time algorithm that achieves an approximation guarantee of m^c, unless $NP = ZPP$. Mu'alem

and Nisan [54] consider the case of *known single-minded bidders*. They show how to combine certain truthful mechanisms into an improved mechanism, while preserving truthfulness. Using this technique, they improve the greedy mechanism of Lehmann et al. by adding a partial enumeration of the space of allocations. The resulting polynomial time mechanism yields an $\epsilon\sqrt{m}$ approximation, for any constant $\epsilon > 0$.

We consider the case of known single-minded bidders when there are $\Omega(\ln K)$ copies of each item available, where K is the maximum size among any of the sets S_i. Each bidder desires only one copy of each good in her desired set; that is, S_i is a set, not a multiset. For the corresponding optimization problem of finding an allocation maximizing total valuation, there is a good approximation algorithm that uses randomized rounding. Our randomized auction mechanism is based on this algorithm. Our auction mechanism runs in polynomial time and attains a $(1+\epsilon)$-approximation to the optimal valuation. The ϵ that we can obtain depends on the constant inside the $\Omega(\ln K)$ bound on the multiplicities.

5.2 Special notation for this chapter

In this chapter, we will depart from the notation set forth in Section 1.3. We will *not* reserve the symbols t_i and q_i for types and allocations. In this chapter, t and q have other meanings, which will be defined when we get to them. Since agent i's type is just her valuation for her desired set, we will denote this by v_i. Since the mechanism is collecting payments from the players and these make up the mechanism's revenue, we will denote these by R_i.

We use the term "profit" instead of "utility." Let the indicator variable $z_i(b)$

be 1 if i wins her desired set and 0 otherwise. Then we define

$$\text{profit}_i(b) = v_i z_i(b) - R_i(b), \tag{5.1}$$

that is, i's valuation for the goods she gets, minus the price she pays. Each agent's goal is to maximize her own profit.

5.3 Our mechanism for known single-minded bidders

We design a randomized mechanism based on solving the natural linear programming relaxation of the SetPacking problem, and randomly rounding the resulting fractional allocation. In the case that the number of copies of each item is $\Omega(\ln K)$ (where K is the maximum size of a set S_i), we prove that our mechanism achieves near-optimal total valuation, is truthful in expectation and strongly truthful with high probability, and has revenue that compares well with a natural variant of VCG. We describe our mechanism by successive refinement of a simple randomized rounding idea, sprinkling in motivation as we encounter and overcome various obstacles.

First recall from Section 2.4.1 that a deterministic mechanism for known single-minded bidders is truthful if and only if the allocation rule is monotone and the price for a winning agent equals her "threshold." That is, if we fix the other bids b_{-i}, then agent i has some threshold bid $T_i(b_{-i})$ such that she wins and pays $T_i(b_{-i})$ if $b_i > T_i(b_{-i})$, and loses if $b_i < T_i(b_{-i})$. (If $b_i = T_i(b_{-i})$, the agent can win and pay $T_i(b_{-i})$, or lose and pay zero; it doesn't matter which.) From Section 2.5 and Theorem 2.4.5, a randomized mechanism is truthful in expectation if and only if for every agent i, the probability $p_i(b_{-i}, b_i)$ that the mechanism assigns her the desired set S_i is increasing in her bid b_i, and her expected payment is equal to a

certain integral of the function p_i:

$$b_i p_i(b_{-i}, b_i) - \int_0^{b_i} p_i(b_{-i}, u)du.$$

Thus, to create a mechanism that is truthful with high probability, it needs to act like a threshold mechanism with high probability, and to get truthfulness in expectation we need the success probabilities of each agent to be increasing in her bid. We work on this second property first.

Our mechanism works as follows. First collect the bids. Using some small fixed $\epsilon' \in (0, 1)$ that is publicly known, pretend that we have only $m'_j = \lfloor (1 - \epsilon')m_j \rfloor$ copies of each item j to distribute. Now solve the following linear program to get an optimal fractional allocation, using the artificially reduced supply of goods.

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{i \in \mathcal{N}} b_i x_i & & \\ \text{subject to:} \quad & \textstyle\sum_{i: j \in S_i} x_i \leq m'_j & \text{for all } j \in \mathcal{G} & \\ & 0 \leq x_i \leq 1 & \text{for all } i \in \mathcal{N} & \end{aligned} \tag{5.2}$$

Denote the optimal fractional allocation by x. We assume that we always find a vertex solution to the linear program, and break ties in a particular fixed way independent of the bids b (e.g., between two vertex solutions, choose the solution with the higher value of x_i for the smallest index i in which they differ). Notice that a fractional value of x_i means that the LP allocates agent i an x_i fraction of *each* good in her set S_i. Now we perform the standard trick of treating the x_i's as probabilities. We define a preliminary set of *initial winners* by selecting each bidder i independently with probability x_i. Unfortunately, we may have tried to sell too many copies of some items, so we will need to modify this outcome by deleting certain selected bidders. The modified outcome will be feasible. However, it is not yet clear how the modification affects the monotonicity of an agent's

overall success probability, so we will have to return to this issue.

First it is not hard to see that, with high probability, no item is over-sold.

Chernoff bound [51]. *Let $X_1, \ldots, X_n$ be independent Poisson trials such that, for $1 \leq i \leq n$, $Pr[X_i = 1] = p_i$. Then for $X = X_1 + \ldots + X_n$, $\mu \geq p_1 + \ldots + p_n$, and any $\alpha < 2e - 1$ we have*

$$Pr[X > (1 + \alpha)\mu] < e^{-\mu\alpha^2/4}.$$

Proposition 5.3.1 *Let $c > 0$. Suppose that each item $j \in \mathcal{G}$ has multiplicity $m_j = \Omega(\ln K)$. Then the probability that a given item is over-sold is at most $\frac{1}{K^{c+1}}$ (the multiplicative constant inside the Ω is $\frac{4(c+1)}{\epsilon'^2(1-\epsilon')}$).*

It is easy to show that this randomized initial allocation is monotone, i.e., that the value x_i in the optimum is monotone in the bid b_i of agent i.

Lemma 5.3.2 *Let x be an optimal solution to a linear program when the objective function vector is b, and let x' be an optimal solution when the objective function vector is b' (where ties are broken independently of b and b'). Suppose $b_i = b'_i$ for all $i \neq i_0$ and $b'_{i_0} > b_{i_0}$. Then either $x' = x$ or $x'_{i_0} > x_{i_0}$.*

Proof: Since x is the optimal solution to the linear program, and x' is a feasible solution,

$$b \cdot x \quad \geq \quad b \cdot x' \tag{5.3}$$

Similarly,

$$b' \cdot x' \quad \geq \quad b' \cdot x \tag{5.4}$$

If $x = x'$, we are already done. Otherwise the two vertices are distinct, and since the tie breaking rule is the same, one of the two inequalities must be strict. Adding

(5.3) and (5.4), and noting that $b' \cdot x = b \cdot x + (b'_{i_0} - b_{i_0})x_{i_0}$ for all vectors x, we get

$$(b'_{i_0} - b_{i_0})(x'_{i_0} - x_{i_0}) > 0$$

from which the result follows. ■

From the above result, it follows that the probability of rounding any agent to 1 is monotone in her bid. Thus, if no items were ever initially oversold, we would get a truthful mechanism. However, there is some probability that an agent i is not allocated her set even though she is initially rounded to 1, because some item j in S_i may be oversold, in which case some or all of the agents desiring j must be dropped. For reasons of egalitarianism (and for lack of a better idea), we propose to drop every agent i whose set S_i contains an oversold item j. As the following example shows, the overall probability that agent i wins is not necessarily monotone in her bid.

Example 1 *Consider an instance where there are 51 agents bidding for an item 'A' with multiplicity 1. Suppose that when the first agent bids a value b_1, the fractional allocation is $\frac{1}{2}$ for agent 1, and $\frac{1}{100}$ for agents $2, \ldots 51$. In this case, the probability that agent 1 is an initial winner is $\frac{1}{2}$, and the probability that she is finally allocated the item is $\frac{1}{2}(1 - \frac{1}{100})^{50} \approx 0.3$.*

Now suppose she raises her bid to $b_2 > b_1$ and the fractional solution changes to 0.51 for agent 1, 0.49 for agent 2, and 0 for all the other agents. In this case, agent 1's probability of being an initial winner increases to 0.51. However, her probability of being allocated the item is now $(0.51)(1 - 0.49) \approx 0.26$. Hence the algorithm described above is not monotone.

As a result, we do not yet have a truthful mechanism. In the next two sections, we show how to achieve monotonicity. We give a very high level description in

Section 5.3.1 and then fill in the necessary details in Section 5.3.2.

5.3.1 Dealing with over-sold items: the basic idea

While the probability that the rounding fails is small, it may depend on the agent's bid and may not be monotone. Our approach is to drop each agent with some additional probability so as to ensure that the overall probability that agent i wins is directly proportional to the variable x_i. When the rounded solution is not feasible, it may still be possible to serve some agents without clashes. We use the following approach:

Step 1. *Solve* the scaled linear programming relaxation (5.2).

Step 2. *Round* each variable x_i to 1 with probability x_i, set to 0 otherwise.

Step 3. *Select* all agents i that are rounded to 1 and such that the supply constraints for all items in S_i are satisfied.

Step 4. *Drop* each agent with some additional probability (to be defined later).

Let $\hat{x}$ denote the integer assignment resulting in Step 2. Consider an agent i_0. The agent is selected in Step 3 if she is rounded to 1 in Step 2 and the constraints for all items in S_{i_0} are satisfied. That is, i_0 is selected if $\hat{x}_{i_0} = 1$ and $\hat{x}$ satisfies

$$\sum_{i:\, j \in S_i,\, i \neq i_0} \hat{x}_i \;\; \leq \;\; m_j - 1 \qquad \text{for all } j \in S_{i_0}.$$

Let $I_{i_0} = \{i : S_i \cap S_{i_0} \neq \emptyset\}$. The variables $x_i : i \in I_{i_0}$ form a feasible solution to the scaled linear program (5.2) induced on the items in S_{i_0}. Let q_{i_0} be the conditional probability that no item in S_{i_0} is over-sold, given that $\hat{x}_{i_0} = 1$. Set $q^* = 1 - \frac{2}{K^c}$. Using Proposition 5.3.1 and the union bound on the items in S_{i_0},

we get that $q_{i_0} > 1 - \frac{1}{K^c} > q^*$. Thus the probability that agent i_0 is selected at Step 3 is $x_{i_0} q_{i_0} > x_{i_0} q^*$. Therefore, in Step 4 we would like to drop agent i_0 with probability $1 - (q^*/q_{i_0})$, so that the probability that agent i_0 survives through the end of Step 4 is exactly $x_{i_0} q^*$. We would then have a monotone allocation algorithm.

5.3.2 Dealing with over-sold items: important details

Note that the algorithm described above requires us to exactly compute the probability q_{i_0}. However, it is #P-hard to compute this number exactly, so the above scheme cannot be implemented efficiently. We get around this problem by using an estimator for this probability. First, we need the following simple observation:

Lemma 5.3.3 *Let x be any vertex of the polytope $\{x : Ax \leq r, 0 \leq x \leq 1\}$, where $A \in \{0,1\}^{m \times n}$ and $r \in \mathbb{Z}^m$. Then $x \in \mathbb{Q}^n$ and each x_i can be written with denominator D, for some $D \leq m!$.*

Proof: Any vertex of the polytope is given by a linear system $\tilde{A}\tilde{x} = \tilde{r}$ where $\tilde{A}$ is a non-singular square submatrix of A, and $\tilde{x}$ and $\tilde{r}$ are corresponding submatrices of x and r respectively. Then $\tilde{x} = \tilde{A}^{-1}\tilde{r}$. Let $D = \det(\tilde{A})$. Since $\tilde{A}$ is a 0-1 matrix, $D \leq m!$. Moreover $D \cdot \tilde{A}^{-1}$ and $\tilde{r}$ are integer matrices and hence $D\tilde{x}$ is an integer vector. Any x_i not in $\tilde{x}$ is set to zero or one, and hence trivially satisfies the conclusion. Hence the claim. ■

Corollary 5.3.4 *Let x', x'' be vertices of the polytope $\{x : Ax \leq r, 0 \leq x \leq 1\}$, where $A \in \{0,1\}^{m \times n}$ and $r \in \mathbb{Z}^m$. Then for each i, either $x'_i = x''_i$ or $x'_i \geq x''_i(1+\delta)$ or $x''_i \geq x'_i(1+\delta)$, where $\delta = (1/m!)^2$.*

Corollary 5.3.4 along with Lemma 5.3.2 imply that whenever an agent i increases her bid, this either has no effect on the allocation or it increases x_i by a factor of at least $(1+\delta)$.

The algorithm described above requires computing $1/q_{i_0}$ for each agent i_0. Instead of the exact value, we use an estimator Y for this number, and in Step 4 we retain agent i_0 with probability q^*Y. Consider the following experiment: Round x_{i_0} to 1. For each $i \in I_{i_0}$, round x_i to 1 independently with probability x_i. Recall that q_{i_0} is defined to be the probability that this solution satisfies the constraints for all items in S_{i_0}. Let the random variable X denote the number of trials of the experiment required before this happens, so that $E[X] = 1/q_{i_0}$. Our estimator Y for $1/q_{i_0}$ is $\min(\frac{1+\delta\epsilon}{N}\sum_{\ell=1}^{N} X^\ell, 1/q^*)$, where $N = O(K^c \log \frac{1}{\delta\epsilon})$, the X^ℓ's are independent trials of the above experiment, c is from Proposition 5.3.1, and $\epsilon \ll 1$ is some error parameter of our choosing. (Later, ϵ will be part of the error probability in our guarantee on truthfulness with high probability. We will typically set $\epsilon \ll 1/K^c$.) The reason we take the min is that in Step 4, we retain agent i_0 with probability q^*Y, so we must ensure that this number is at most 1. The expectation of $\frac{1+\delta\epsilon}{N}\sum_{\ell=1}^{N} X^\ell$ is exactly $(1+\delta\epsilon)/q_{i_0}$, which is less than and bounded away from $1/q^*$. Since $\sum_{\ell=1}^{N} X^\ell$ is a negative binomial random variable with success probability $q_{i_0} \approx 1$, its distribution is concentrated about its mean, so the expectation of Y will not be much smaller than $(1+\delta\epsilon)/q_{i_0}$. In particular, we can use moment generating functions to bound the upper tail's contribution to the expectation (see e.g., [51]), and obtain the following.

Lemma 5.3.5 *The estimator Y defined above is at most $1/q^*$, and*

$$E[Y] \in [1/q_{i_0}, (1+\delta\epsilon)/q_{i_0}]$$

.

This finishes the description of our allocation algorithm, which we will call RandRound. Note that the probability that agent i_0 is not dropped in Step 4 of the algorithm above is exactly $q^*E[Y]$. We now argue that this mechanism is monotone.

Theorem 5.3.6 *The probability that an agent i is selected by the algorithm* RandRound *is monotone increasing in her bid b_i.*

Proof: Fix an agent i and a vector of bids b_{-i} for agents other than i. Let $b'_i > b_i$. Consider the corresponding LP optima x and x'. From Lemma 5.3.2 either $x = x'$ or $x'_i > x_i$. In the first case, the experiment is the same and hence agent i's probability of succeeding is the same whether she bids b_i or b'_i. In the second case, her probability $p_i(b_{-i}, b_i)$ of winning if she bids b_i is given by

$$\begin{aligned} p_i(b_{-i}, b_i) &= x_i q_i q^* E[Y] \\ &\leq x_i q_i q^* (1 + \delta\epsilon)/q_i \\ &= x_i q^* (1 + \delta\epsilon) \end{aligned}$$

If she bids b'_i, then her probability of winning is

$$\begin{aligned} p_i(b_{-i}, b'_i) &= x'_i q'_i q^* E[Y'] \\ &\geq x'_i q'_i q^* / q'_i \\ &\geq x_i q^* (1 + \delta) \end{aligned}$$

where the last inequality follows from Corollary 5.3.4. Since $\epsilon < 1$, this shows that the above LP rounding algorithm is monotone. ■

Theorem 5.3.7 *Suppose that each item has multiplicity $\Omega(\ln K)$, as in Proposition 5.3.1. Let OPT denote the optimal total valuation achievable by any alloca-*

tion. Then the expected total valuation achieved by the above algorithm is at least $(1-\epsilon')q^*OPT$.

Proof: Every feasible allocation gives a feasible solution to the LP with the actual multiplicities m_j. Scaling down any solution to this LP by a factor of $(1-\epsilon')$ yields a solution to our LP (5.2) with the artificially reduced multiplicities m'_j. Therefore, the optimal solution to this LP has value $\sum_{i\in\mathcal{N}} b_i x_i \geq (1-\epsilon')OPT$. The probability that agent i is selected is at least $x_i q^*$, hence the expected total valuation is at least $(1-\epsilon')q^*$ times the LP optimum. ■

5.4 Computing payments

We have succeeded in creating an allocation rule where the success probability for i increases monotonically with her bid. We now consider how to compute payments in order to obtain a truthful mechanism. Recall that $p_i(b_{-i}, b_i)$ denotes the overall probability that i wins her desired set S_i. Note that b_{-i} is fixed throughout this discussion, so we suppress it in the notation where convenient. Recall from Section 2.5 and Theorem 2.4.5 that to guarantee truthfulness in expectation, i's expected payment should be

$$R_i(b_{-i}, b_i) = p_i(b_{-i}, b_i)b_i - \int_0^{b_i} p_i(b_{-i}, u)du. \tag{5.5}$$

Notice that $p_i(b_{-i}, u)$ is a step function that jumps whenever the selected vertex in the LP (5.2) changes, and is flat elsewhere. See Figure 5.1. Thus, it could have exponentially many breakpoints, so we cannot compute R_i directly. However, we can use the trick from Section 2.6 of sampling from the bid axis to achieve the correct expected payments, which yields a mechanism that is truthful in expectation.

The goal of this section is to create a payment scheme that simultaneously

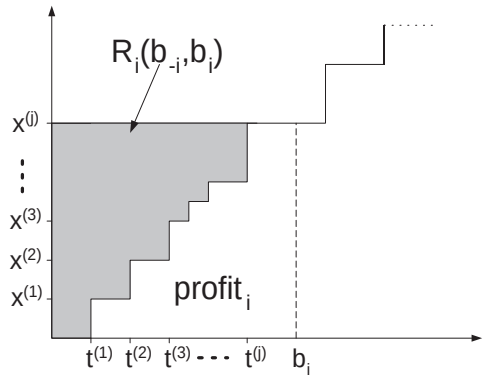

Figure 5.1: The graph shows i's probability $p_i(b_{-i}, b_i)$ of winning as a function of her bid b_i. The gray area $R_i(b_{-i}, b_i)$ is her expected payment. If b_i is a truthful bid, then the white area is her expected profit.

gives us truthfulness with high probability and in expectation. In Section 5.4.1, we show how to attain truthfulness with high probability (but not in expectation), using a simpler (non-monotone) allocation rule. In Section 5.4.2, we show how to combine this payment scheme with our monotone allocation rule of Section 5.3.1 to simultaneously obtain truthfulness in expectation and with high probability.

5.4.1 Threshold payments

Consider the simpler allocation rule where we leave out Step 4 (the drop step). As previously noted, this allocation may not be monotone, in which case there is no payment scheme that is truthful in expectation. Here we give payments that are instead truthful with high probability. The idea here is very similar to that of sampling from the quantity axis, as in Section 2.6

In the bidder selection step, let us perform the rounding by selecting n independent uniform$[0, 1]$ random variables $y_1, \ldots, y_n$, and choosing i to be an initial winner if $y_i \leq x_i$. Note that each bidder i is selected by this experiment indepen-

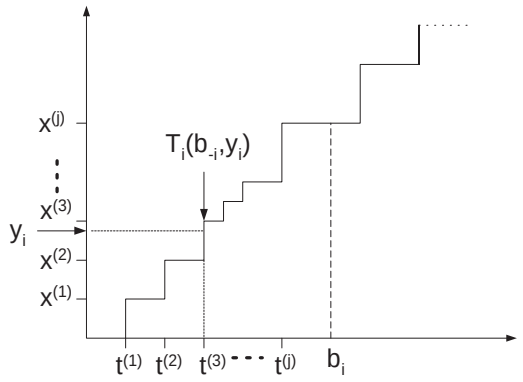

Figure 5.2: The graph shows i's fractional allocation as a function of her bid b_i (with b_{-i} fixed). It is a step function that is flat while one vertex of the LP (5.2) stays optimal, then jumps when another vertex becomes preferred. Since y_i lands in $(x^{(2)}, x^{(3)})$, i's payment is $t^{(3)}$.

dently with probability x_i, as required by Step 2. For each winning bidder i we compute a price that will depend on the outcome of the random variable y_i (and of course also on b_{-i}). Fix b_{-i} and a realization of the random cutoff y_i. There is a *threshold* value $T_i(b_{-i}, y_i)$ such that i will lose if $b_i < T_i(b_{-i}, y_i)$ and be an initial winner if $b_i > T_i(b_{-i}, y_i)$. See Figure 5.2. This threshold is the point at which $x_i(b_{-i}, b_i)$, considered as a function of b_i with b_{-i} fixed, first rises to y_i. We set agent i's price to be $T_i(b_{-i}, y_i)$ if she actually wins.

To compute $T(b_{-i}, y_i)$, we binary search on b_i, re-solving the LP each time. For the vector of bids $(b_{-i}, T_i(b_{-i}, y_i))$ there are two equally good fractional allocations x and x'. Therefore, $T_i(b_{-i}, y_i)(x_i - x'_i) = \sum_{j \neq i} b_j(x'_j - x_j)$. Assuming all bids are given to d bits of precision, we can express $T_i(b_{-i}, y_i)$ as a fraction with denominator at most $2^d(m!)^4$ (by Lemma 5.3.3), so we can use binary search and the method of Diophantine approximation to compute it exactly in polynomial time (see e.g., [66]).

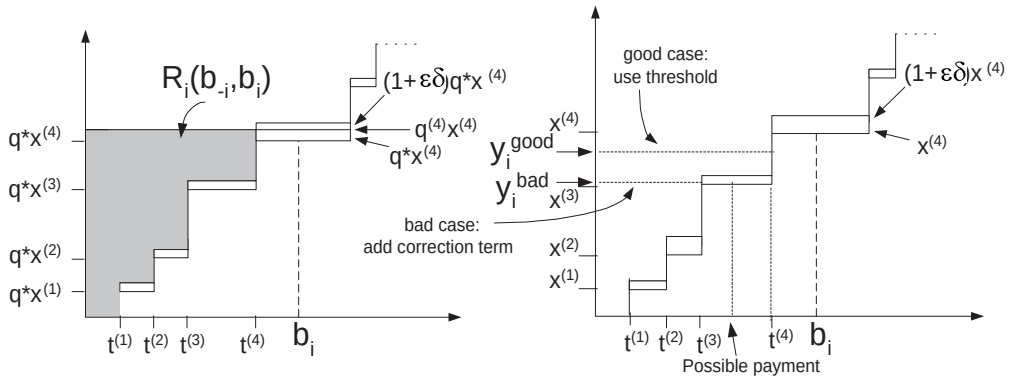

Figure 5.3: The left graph shows agent i's probability of success as a function of her bid b_i. The boxes indicate our margin of uncertainty about this probability. Modulo this uncertainty, the shaded area denotes the truthful payment function. The right graph illustrates our payment scheme.

If our mechanism never had to throw away any initial winners, then our allocation algorithm would be universally truthful. Suppose we fix a particular realization ω of the vector of random variables $y_1, \ldots, y_n$. Then the only circumstance under which agent i could benefit by lying is if i is selected as an initial winner, but is discarded because one of the items in S_i is oversold. Since this probability is at most $\frac{1}{K^c}$, our mechanism is truthful with high probability.

5.4.2 Combining threshold payments with the monotone allocation

We now show how to modify this threshold scheme to get truthfulness in expectation, using the monotone allocation rule of Section 5.3.1. If it were the case that $p_i = q^* x_i$ for all i, i.e., each agent's probability of winning were directly proportional to her fractional allocation from the LP, then the threshold payment scheme

would give the correct expected payment, so it would already be truthful in expectation. The problem is that we just have $p_i = q_i^* x_i$ for some $q_i^* \in [q^*, q^*(1+\delta\epsilon)]$, and moreover we cannot compute the q_i^* exactly. Our solution is to use the threshold payments as a first approximation, then add a small correction on a set of small probability.

Let the fractional solution values (the steps in Figure 5.2) be $x^{(1)}, x^{(2)}, \ldots, x^{(j)}$ such that the solution for agent i's actual bid b_i is $x^{(j)}$. Moreover, let $q^{(k)}$ be the probability that the sale to agent i survives Steps 3 and 4 given that she is selected in Step 2, when the solution corresponding to $x^{(k)}$ is used. Thus, her overall success probability is $q^{(k)}x^{(k)}$ in this case. To get truthfulness in expectation, when agent i wins, her expected payment should be

$$\frac{1}{q^{(j)}x^{(j)}}\left(b_i x^{(j)} q^{(j)} - \sum_{k=1}^{j-1}(t^{(k+1)} - t^{(k)})q^{(k)}x^{(k)} - (b_i - t^{(j)})q^{(j)}x^{(j)}\right).$$

Suppose we use the threshold payment scheme. Given that agent i wins, y_i is distributed uniformly on $[0, x^{(j)}]$. Thus, agent i's expected payment is

$$\frac{1}{x^{(j)}}\sum_{k=1}^{j} t^{(k)}(x^{(k)} - x^{(k-1)}),$$

which rearranges to

$$\frac{1}{x^{(j)}}\left(x^{(j)}b_i - \sum_{k=1}^{j-1}(t^{(k+1)} - t^{(k)})x^{(k)} - (b_i - t^{(j)})x^{(j)}\right)$$

where $x^{(0)} = 0$. Therefore, we must add some correction term to increase this payment by

$$\sum_{k=1}^{j-1}(t^{(k+1)} - t^{(k)})\frac{x^{(k)}}{x^{(j)}}\left(1 - \frac{q^{(k)}}{q^{(j)}}\right)$$

in expectation.

One way to do this is to add $(t^{(k+1)} - t^{(k)})\frac{1-q^{(k)}/q^{(j)}}{\epsilon\delta}$ whenever $y_i \in [x^{(k)}, (1+\epsilon\delta)x^{(k)}]$, for $k = 1, \ldots, j-1$. See Figure 5.3. Since we do not know $q^{(k)}$ and

$1/q^{(j)}$, we must replace them in the formula with independent unbiased estimators. These estimators can be obtained by running our allocation algorithm. The expected payment is now that given by formula (5.5), so the mechanism is truthful in expectation. When $y_i \notin [x^{(k)}, x^{(k)}(1+\epsilon\delta)]$ for all k, our payments are just threshold payments. This happens with probability at least $(1-\epsilon)$ since by Corollary 5.3.4, consecutive fractional allocations $x^{(k)}$ are spaced by factors of at least $(1+\delta)$. As we argued in the last section, the threshold mechanism is strongly truthful with high probability. Hence we get the following. (Recall $\epsilon \ll 1/K^c$.)

Theorem 5.4.1 *Assuming the item multiplicities are all $\Omega(\ln K)$ as in Proposition 5.3.1, allocation algorithm* RandRound *combined with the above payment scheme is truthful in expectation and is also strongly truthful with error probability $\epsilon + \frac{1}{K^c}$.*

5.5 Revenue considerations

Now we consider the revenue generated by our auction. We show that the expected revenue generated is very close to that generated by a natural fractional relaxation of the VCG mechanism. First we must define this mechanism.

Recall the VCG mechanism: it chooses a feasible allocation that maximizes the *utilitarian objective function*, which is the total reported valuation of the winning agents. That is, it selects some

$$x^*(b) \in \operatorname{argmax}_x \sum_{i \in \mathcal{N}} b_i x_i, \tag{5.6}$$

where x runs over all feasible allocations.

The mechanism computes a bonus for each bidder, based on that bidder's *marginal value*, which is the difference she made in the objective function by par-

ticipating. Formally, let $V(\mathcal{N}-\mathcal{N}')$ denote the maximum total valuation achievable in (5.6) when the agents in $\mathcal{N}'$ are removed. Then bidder i's marginal value is defined to be $V(\mathcal{N}) - V(\mathcal{N}-i)$. The mechanism charges $R_i(b) = b_i x_i - (V(\mathcal{N}) - V(\mathcal{N}-i))$ to each agent i. (This formula evaluates to zero for agents who lose.) The utilitarian allocation is clearly monotone. Moreover, the objective function is indifferent about satisfying bidder i when she bids exactly her threshold. So if she wins, then $V(\mathcal{N}) - V(\mathcal{N}-i) = b_i - T_i(b_{-i})$, so the VCG payment is $T_i(b_{-i})$. Notice that if agent i bids truthfully, then her profit is equal to her marginal value.

The VCG mechanism is defined with respect to a set of feasible allocations. Usually we maximize over all feasible integer allocations, meaning that each bidder either wins or loses, and no item is over-sold. However, we could consider enlarging the set of allowed allocations to permit fractional allocations. That is, we could allow ourselves to let agent i win to a fractional extent x_i, which would mean that she receives an x_i fraction of each good in S_i. In other words, we would be maximizing the linear program (5.2), using the actual multiplicities m_j. Of course, we could implement such a mechanism only if the goods were divisible. We assume that agent i attains a benefit of $v_i x_i$ from winning to the fractional extent x_i. Thus, she wishes to maximize $\text{profit}_i(b) = v_i x_i(b) - R_i(b)$. Agent i's marginal value to the system is $V(\mathcal{N}) - V(\mathcal{N}-i)$, where $V(\mathcal{N}')$ is the optimal LP value using only the agents in $\mathcal{N}'$. The VCG payment formula becomes $R_i(b) = b_i x_i - (V(\mathcal{N}) - V(\mathcal{N}-i))$. We refer to this resulting mechanism as *fractional VCG*, or FVCG for short.

Similarly, we can define an FVCG mechanism with respect to the artificially reduced multiplicities m'_j. We will show that the expected revenue of our mechanism is almost the same as the revenue generated by the fractional VCG mechanism

using the reduced multiplicities.

First we obtain an expression for the revenue generated by the fractional VCG mechanism. Fixing b_{-i}, how does the optimal allocation $x_i(b_{-i}, b_i)$ change as i increases her bid from 0 upward? Suppose the mechanism always selects some vertex solution of the LP. Initially, $x_i = 0$. The only part of the LP that changes is the direction of the objective function vector, not the polytope of feasible solutions. Thus, the optimal x_i remains zero for an interval until i's bid hits some threshold $t^{(1)}$. At this point, some other vertex solution with $x_i = x^{(1)} > 0$ becomes optimal. Now this solution remains optimal for some interval, until x_i jumps again at $b_i = t^{(2)}$ to some higher level $x^{(2)}$. Suppose x_i jumps j times at $t^{(1)}, \ldots, t^{(j)}$ to new levels $x^{(1)}, \ldots, x^{(j)}$ as we raise i's bid to its actual value b_i. For bids in $(t^{(k)}, t^{(k+1)})$, the LP value increases at rate $x^{(k)}$. Thus, i's marginal value is $\sum_{k=1}^{j-1} x^{(k)}(t^{(k+1)} - t^{(k)}) + x^{(j)}(b_i - t^{(j)})$, and her price R_i is $b_i x_i$ minus this, which is

$$\sum_{k=0}^{j-1}(x^{(j)} - x^{(k)})(t^{(k+1)} - t^{(k)}),$$

where $x^{(0)} = t^{(0)} = 0$. To visualize this computation consider the Figure 5.1, with the curve denoting $x_i(b_{-i}, b_i)$ (whereas originally the curve in Figure 5.1 was the probability of being selected as a winner, which is $\approx q^* x_i(b_{-i}, b_i)$).

In our mechanism, the expected payment by agent i is

$$\sum_{k=0}^{j-1}(q^{(j)}x^{(j)} - q^{(k)}x^{(k)})(t^{(k+1)} - t^{(k)}),$$

where $q^{(k)}$ is the probability that the sale to agent i is not cancelled in Steps 3 and 4, if i were to bid between $t^{(k)}$ and $t^{(k+1)}$ (just as in Section 5.4.2). Thus, we are comparing vertical strips of equal width, and height $x^{(j)} - x^{(k)}$ for FVCG as

opposed to height $q^{(j)}x^{(j)} - q^{(k)}x^{(k)}$ for our mechanism. But

$$\begin{aligned} q^{(j)}x^{(j)} - q^{(k)}x^{(k)} &\geq q^*x^{(j)} - (1+\epsilon\delta)q^*x^{(k)} \\ &\geq (1-\epsilon)q^*(x^{(j)} - x^{(k)}) \end{aligned}$$

because $q^{(k)} \in [q^*, q^*(1+\epsilon\delta)]$ and $x^{(j)} \geq (1+\delta)x^{(k)}$ for all $k < j$.

Theorem 5.5.1 *Suppose that each item $j \in \mathcal{G}$ has multiplicity $m_j = \Omega(\ln K)$, as in Proposition 5.3.1. Then the expected revenue generated by* RandRound *is at least $(1-\epsilon)q^*$ times the revenue generated by the FVCG mechanism with multiplicities m'_j.*

Under the same conditions on the multiplicities, the probability that agent i actually wins is also at least q^*x_i. Thus, the auction essentially implements the FVCG mechanism on the artificially reduced multiplicities.

5.5.1 Comparing against the "optimal" mechanism

It is natural to ask how our revenue compares with that of an "optimal" truthful mechanism, but it turns out that even posing this question correctly is a tricky endeavor. One truthful mechanism is to arbitrarily select a feasible set $\mathcal{W}$ of possible winners, set fixed prices P_i for every bidder in that set, and refuse to sell to any other agents. Any agent i with $v_i \geq P_i$ will then buy her set at price P_i. If we happen to get lucky and choose $\mathcal{W}$ to be the feasible set of bidders that maximizes the total valuation, and happen to choose $P_i = v_i$ for each $i \in \mathcal{W}$, then we reap the entire valuation as revenue. However, this "omniscient mechanism" hardly seems a fair benchmark. In fact, it is well-known that even when auctioning just a single copy of a single item, no truthful mechanism can always attain a guaranteed

fraction of the optimal valuation, because there is no way to deal with a single astronomical bidder.

Therefore, in the single item case, [28, 23, 40] suggest comparing against variants of the VCG mechanism. We have shown that our auction achieves expected revenue approximately equal to that of the FVCG mechanism with a slightly reduced supply of goods. One can construct a pair of examples showing that neither the VCG nor the FVCG mechanism's revenue dominates the other. Moreover, it is well-known that artificially decreasing the supply of goods can sometimes dramatically increase revenues. (See [28] for a striking example.) Therefore, it is unclear how the revenue compares with that of the VCG mechanism using the full supply. This could be an interesting direction for further work. A huge body of literature addresses how various auctions fare under probabilistic assumptions about the valuations, but we are not aware of any such work applied to combinatorial auctions. This is another interesting future direction.

5.6 Lying about the set: an example

It is natural to ask if we can extend our method to handle the case where the set S_i is part of agent i's bid (i.e., the case of single-minded bidders, instead of *known* single-minded bidders). The following example shows that, if we insist that agent i wins with probability roughly proportional to the fractional allocation x_i given by the LP, then it is impossible to obtain a mechanism for single-minded bidders that is strongly truthful with high probability.

Suppose there are three items $\{A, B, C\}$ and three bidders $\{1, 2, 3\}$. One copy of each item is available. Suppose bidder 1 bids 2 units for set $\{B, C\}$ (the truth), bidder 2 bids $\frac{3}{2}$ units for set $\{A, B\}$ and bidder 3 bids $\frac{3}{2}$ units for set $\{A, C\}$.

Then the LP solves to $x = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ (with total valuation $\frac{5}{2}$). If bidder 1 lies by increasing her set to $\{A, B, C\}$ (but still bidding 2 units for it), then the LP solves to $x = (1, 0, 0)$ (with total valuation 2). Suppose that we actually had three copies of each item available, but were just using 1 as the reduced multiplicities. Then no item is ever over-sold. Thus, our mechanism (without the drop step) implements fractional VCG in expectation, so is still truthful in expectation. However, we show that with probability $\frac{1}{2}$, bidder 1 benefits by lying, and hence we do not have strong truthfulness with high probability.

When agent 1 bids her true set, her probability $p_1(b_{-1}, b_1)$ of winning stays constant at $\frac{1}{2}$ for $b_1 \in (0, 3]$. Thus the threshold payment scheme of section 5.4.1 would charge her a price of 0 (leading to a profit of 2) for any $y_1 \in [0, \frac{1}{2}]$. When y_1 falls in $(\frac{1}{2}, 1]$, however, she loses and has a profit of 0.

On the other hand, if she lies about her set as above, then $p_1(b_{-1}, b_1)$ jumps from 0 to 1 at $b_1 = \frac{3}{2}$. Thus, with probability 1, she wins and pays $\frac{3}{2}$, so her net profit is $\frac{1}{2}$, irrespective of y_1. Note that when her rounding variable y_1 lands in $(\frac{1}{2}, 1]$ (which happens with probability $\frac{1}{2}$), her profit when she lies is more than her profit when she tells the truth. Thus with probability $\frac{1}{2}$, the mechanism is not universally truthful.

We can extend this example to arbitrarily high item multiplicities by simply adding in appropriate bidders j who bid high enough that they are fully satisfied (i.e., the optimal solution has $x_j = 1$). Note that in this case the reduced multiplicities are smaller than the actual multiplicities only by an additive -2, not a multiplicative $\frac{1}{3}$.

5.7 Conclusion

We have shown a general technique to modify a linear program rounding algorithm to make it monotone. This gives an approximately efficient truthful mechanism (in expectation and with high probability) for the combinatorial auction problem with single parameter agents. The simple rounding algorithm can be derandomized using pessimistic estimators [61]. It would be interesting to see if the algorithm can be derandomized while maintaining its monotonicity.

Chapter 6

Collusion

Truthful dominant strategy mechanisms are often called *strategyproof* because no single agent can succeed in cheating the mechanism by playing any strategy other than truth-telling. However, this does not rule out the possibility of collusion by two or more agents to help each other cheat the mechanism. Mechanisms that also guard against this kind of manipulation are called *group strategyproof*. Although there is one generic method for creating group strategyproof mechanisms for certain cost-sharing problems [52, 53], this is generally a very difficult property to ensure, and other examples in the literature of mechanisms with this property are rare.

In particular, VCG mechanisms are in general *not* group strategyproof. If we want to use a VCG mechanism in a situation where we suspect agents might be able to collude, it is important to understand which agents could successfully team up to defeat the mechanism, and under what circumstances this can occur. For some simple VCG mechanisms, such as the single-item Vickrey auction, this is easy to understand. But for other mechanisms, the intricacies of group manipulations are far less obvious.

In this chapter, we consider multicast cost sharing, a much-studied model in algorithmic game theory. We investigate the group manipulability properties of one interesting mechanism, the *marginal cost* (MC) mechanism (proposed by Moulin and Shenker [53]). This mechanism is a member of the VCG family, and thus is truthful and maximizes the utilitarian objective, but it is not group strategyproof. We completely characterize the circumstances under which a group can successfully manipulate the mechanism.

6.1 Definitions

Recall that our definition of a truthful mechanism in Section 1.3 is that truth-telling is a *weakly* dominant strategy for each agent. Thus, even though $u_i(b_{-i}, v_i) \geq u_i(b_{-i}, b_i)$, for all i, v_i, b_{-i}, and b_i, there could be some $b_i \neq t_i$ where equality holds. That is, for certain values of b_{-i}, there may be some lying bids that i can make without hurting herself. In so doing, i may be able to help other agents. In other words, strategyproofness does not preclude the possibility of a coalition of users colluding to improve their utilities.

Any reported valuation profile b can be considered a group strategy for any group $S \supseteq \{i \mid b_i \neq v_i\}$. It will be handy to have a notation for perturbing reported valuations. If b is one valuation profile, and $\hat{b}_S$ is a vector of valuations for agents in the set S, then let $b|^S\hat{b}_S$ denote the vector whose i^{th} component is b_i if $i \notin S$ and $\hat{b}_i$ if $i \in S$. Thus, if S is the strategizing set, we can write the reported valuation profile as $v|^S b_S$. A mechanism $\mathcal{M}$ is *group-strategyproof* (GSP) if there is no group strategy such that at least one member of the strategizing group improves his utility while the rest of the members do not reduce their utilities. In other words, if $\mathcal{M}$ is GSP, the following property holds for all v, b, and $S \supseteq \{i|v_i \neq b_i\}$:

$$\text{either} \quad u_i(b) = u_i(v) \ \forall i \in S$$

$$\text{or } \exists i \in S \text{ such that } u_i(b) < u_i(v).$$

Notice that this definition of group strategyproofness assumes that the agents are not able to exchange money as part of their collusion. One could make an alternative stronger definition that no group of agents can improve their total utility by cheating, in which case the mechanism would be resistant even to coalitions who can exchange money afterwards to compensate a member who "took a hit for the

team." However, we will stick to our weaker definition, which is more standard in the literature.

The single item Vickrey auction described in Section 1.2.1 is clearly not group strategyproof. Recall that this auction awards the item to the highest bidder, at a price equal to the second-highest bid. Because the agent with the second-highest valuation can lower her bid in order to lower the price that the highest bidder must pay, these two bidders alone can successfully manipulate the mechanism (provided the third-highest valuation is strictly lower than the second-highest). Because the structure of this mechanism is so simple, it is trivial to characterize the sets of bidders who can successfully collude, and how they can do it. In particular, any set containing the bidder with the top valuation and all bidders tied for the second-highest valuation has a successful group strategy that consists of lowering the bids such that the bidder with the highest valuation still wins, but at a lower price.

The MC mechanism for multicast cost-sharing is more complicated, and so the analysis of successful group manipulations is considerably more involved.

6.2 The multicast cost-sharing problem

Multicast routing is a technique for transmitting a packet from a single source to multiple receivers without wasting network bandwidth. To achieve transmission efficiency, multicast routing constructs a directed tree that connects the source to all the receivers and sends only one copy of the packet over each link of the directed tree. When a packet reaches a branch point in the tree, it is duplicated and a copy is sent over each downstream link. Multicasting large amounts of data to large groups of receivers is likely to incur significant costs, and these costs need to be covered by payments collected from the receivers. However, receivers cannot

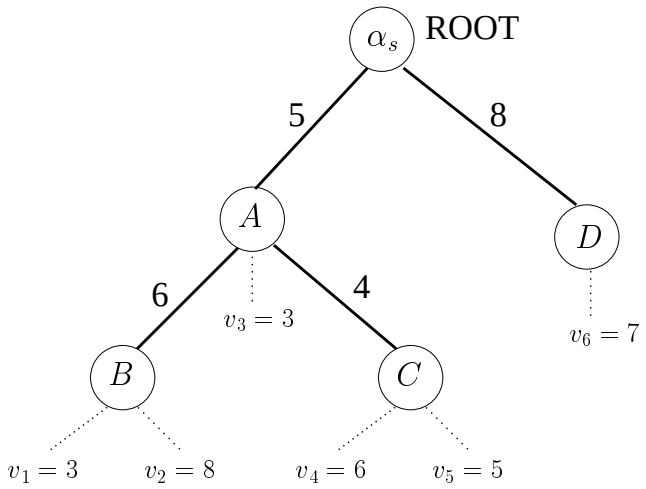

Figure 6.1: A multicast cost-sharing problem.

be charged more than what they are willing to pay, and the transmission costs of shared network links cannot be attributed to any single receiver. Thus, one must design cost-sharing mechanisms to determine which users receive the transmission and how much they are charged.

Figure 6.1 depicts an instance of the multicast cost-sharing problem. There are five potential receivers, each located at a particular node of the multicast tree and each having a certain valuation for receiving the multicast transmission. For example, the notation $v_1 = 3$ beside the leftmost node on the second level from the top means that potential receiver number 1 is located at this node and is willing to pay at most 3 to receive the transmission. The numerical values on the links represent the costs of sending the transmission over those links. The source of the transmission is the root node at the top level of the tree. If $R \subseteq \{1, \ldots, 6\}$ is the set of actual receivers, then the transmission will be sent only to the nodes of the tree at which members of R are located. The total cost of this transmission will be the sum of the costs of the links in the smallest subtree that contains

these nodes and the root. For example, if $R = \{2, 3, 4\}$, then the total cost of the transmission would be 15. The role of a *cost-sharing mechanism* is to determine, for each instance, what the receiver-set R should be and how much each member of R should be charged.

The multicast cost-sharing problem has been studied extensively in recent years, first from a networking perspective [34], then from a mechanism-design perspective [53], and most recently from an algorithmic perspective [22, 20, 1, 38, 23]. Computationally efficient cost-sharing algorithms are desirable because the computational resources of the multicast infrastructure (i.e., link bandwidth and nodes' memory and CPU cycles) must be used to compute them; the point of this infrastructure is to deliver content efficiently, not to do cost-sharing, and hence the latter must not consume enough resources to interfere with the former. The MC mechanism is attractive because it can be computed very simply in a distributed fashion, with just two small messages per link and very simple computations at the nodes. Other cost-sharing methods in the literature use considerably more computation [22].

Formally, the *multicast cost-sharing mechanism design* problem involves an agent population $\mathcal{N}$ residing at a set of network nodes N that are connected by bidirectional network links L. The multicast flow emanates from a source node $\alpha_s \in N$; given any set of receivers $R \subseteq \mathcal{N}$, the transmission flows through a *multicast tree* $T(R) \subseteq L$ rooted at α_s and spanning the nodes at which agents in R reside. It is assumed that there is a *universal* tree $T(\mathcal{N})$ and that, for each subset $R \subseteq \mathcal{N}$, the multicast tree $T(R)$ is merely the smallest subtree of $T(\mathcal{N})$ required to reach the elements in R.[1]. Since we usually draw the tree with the root α_s at the

[1]This approach is consistent with the design philosophy embedded in essentially all current multicast-routing proposals (see, e.g., [8, 17, 18, 36, 60])

top, if node α lies along the path from node β to α_s (and $\alpha \neq \beta$) then we say α lies *above* β, and is an *ancestor* of β. Symmetrically, β lies *below* α and is a *descendent* of α. If these two nodes are directly connected by a link, then α is β's *parent*, and β is a *child* of α. Each link $l \in L$ has an associated cost $c(l) \geq 0$ that is known by the nodes on each end, and each agent i assigns a valuation $v_i \geq 0$ to receiving the transmission. Let $v = (v_1, v_2, \ldots, v_{|P|})$ denote the vector of valuations. Only agent i knows her true valuation v_i.

A cost-sharing mechanism determines which agents receive the multicast transmission and how much each receiver is charged. Since the agents' valuations are private information, the mechanism will ask each agent to report some valuation b_i, and base its decisions on the input vector b of these reported valuations. We let $P_i(b) \geq 0$ denote how much agent i is charged and $q_i(b)$ denote whether agent i receives the transmission; $q_i(b) = 1$ if the agent receives the multicast transmission, and $q_i(b) = 0$ otherwise. The mechanism $\mathcal{M}$ is then a pair of functions $\mathcal{M}(b) = (q(b), P(b))$. The *receiver set* for a given input vector is $R(b) = \{i \mid q_i(b) = 1\}$. An agent's individual utility is given by the quasilinear form $u_i(b) = q_i(b)v_i - P_i(b)$. Notice that $u_i(b)$ does depend on i's true valuation v_i, but we suppress this in the notation. The cost of the tree $T(R)$ reaching a set of receivers R is $c(T(R))$, and the *net worth*, is $NW(R) = v_R - c(T(R))$, where $v_R = \sum_{i \in R} v_i$ and $c(T(R)) = \sum_{l \in T(R)} c(l)$. The overall net worth measures the total benefit of providing the multicast transmission (the sum of the valuations minus the total transmission cost). Of course, the mechanism does not have direct access to v, so it can only compute $NW_b(R) = b_R - c(T(R))$, the net worth with respect to the reported valuations.

The MC mechanism, a member of the VCG family defined in Section 2.3.2,

chooses the receiver set R that maximizes $NW_b(R)$. Let W_b denote this maximum net worth $NW_b(R)$. For each $i \in R$, let W_b^{-i} be the net worth of the receiver set that the MC mechanism would have computed if i had not participated (i.e., if b_i had been set to 0). Then $W_b - W_b^{-i}$ measures the gain in overall net worth that results from i's participation. The cost share that MC assigns to i is $P_i(b) \equiv b_i q_i(b) - (W_b - W_b^{-i})$. Notice that MC satisfies the voluntary participation property ($u_i(b) \geq 0$ so long as i bids truthfully), and never pays an agent to receive the transmission.

For the instance shown in Figure 6.1, when all agents bid truthfully, the MC mechanism computes the receiver set R to be $\{1, 2, 3, 4, 5\}$, resulting in a total transmission cost of 15 and net worth of 10. In this case, agent 2 pays $P_2 = 3$, and all other agents pay zero.

A cost-sharing mechanism satisfying the property that the sum of the prices paid by the agents exactly covers the cost of the multicast tree is said to be *budget-balanced*. As we have just seen, the MC mechanism does not guarantee budget-balance. In the example above, the cost shares of the agents in the MC mechanism sum to 3, not covering the total cost of 15. We view it as a bit of a misnomer to refer to such a mechanism as cost-sharing, since the cost-shares do not come close to covering the entire cost. However, this is the common term used in the literature, so we stick to it.

In [22], a distributed algorithm is given that computes the MC receiver set and cost shares by sending just two modest-sized messages over each $l \in L$ and doing two very simple calculations at each node. We review this algorithm in the next section, before moving on to characterize the groups that can strategize successfully against the MC mechanism, and the conditions under which they can

do so.

6.3 Group strategies that succeed against the MC mechanism

Since the MC mechanism is an instance of the VCG family, it is strategyproof, but not necessarily group strategyproof. In this section, we characterize exactly how the MC mechanism fails to be group strategyproof.

We say that a strategy b for a group S is a *successful group strategy at valuation profile* v if

- $S \supseteq \{i \in P | b_i \neq v_i\}$,
- $\forall i \in S,\ u_i(b) \geq u_i(v)$, and
- $\exists j \in S$ such that $u_j(b) > u_j(v)$.

In other words, a successful group strategy at v is one that (compared to truth-telling) harms none of the members of the group and benefits at least one. Note that the member who benefits may not be one of the members who misrepresented her valuation. If the group S has only two members, we call the strategy a *successful pair strategy*. If there is no group that has a successful strategy at v, then we say that the mechanism is *GSP at* u. A GSP mechanism is one that is GSP at all v.

It is well known that the MC mechanism is not GSP. However, it is not obvious in general which forms of collusion would result in successful manipulation. In this section, we examine this issue in detail by asking two questions. First, at which valuation profiles is MC GSP? Second, for a valuation profile v at which MC is

not GSP, exactly which groups can strategize successfully? We will show that MC fails to be GSP at v if and only if there exists a successful pair strategy or a specific simple kind of three-agent strategy, and show exactly when these strategic opportunities arise.

Feigenbaum et al. [22] give a low network complexity algorithm for the MC mechanism. The algorithm itself gives insights into the workings of the mechanism, so we describe it here.

Given a reported valuation profile b, the receiver set is the unique *maximal efficient set* of agents. To find it, we recursively compute the *worth* $W_b(\beta)$ of each node $\beta \in N$ as

$$W_b(\beta) = \left(\sum_{\substack{\gamma \in \mathrm{Ch}(\beta) \\ W_b(\gamma) \geq 0}} W_b(\gamma) \right) - c(l) + b(\beta)$$

where $\mathrm{Ch}(\beta)$ is the set of children of β in the tree, $c(l)$ is the cost of the link connecting β to its parent node, and $b(\beta)$ denotes the total reported valuation of the agents residing at β. The worth of a node β measures the marginal amount that this node and the optimal subtree below it would contribute to the net worth of the chosen multicast tree, assuming that all the nodes above β had already been included. We can easily compute the worth at the leaves of the tree, then work our way up the tree to compute $W_b(\cdot)$ for the remaining nodes recursively. The maximal efficient set $R(b)$ is the set of all agents i such that every node on the path from i to the root has nonnegative worth.

Another way to view this is as follows: The algorithm partitions the universal tree $T(\mathcal{N})$ into a forest $F(b) = \{T_1(b), T_2(b), \ldots, T_k(b)\}$. A link from $T(\mathcal{N})$ is included in the forest if and only if the child node has nonnegative worth. This is illustrated in Figure 6.2. $R(b)$ is then the set of agents at nodes in the subtree

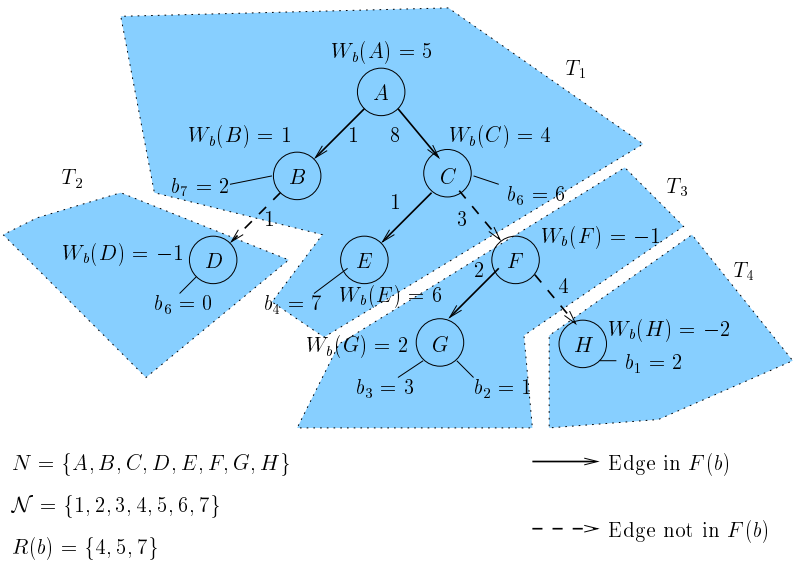

Figure 6.2: Forest induced by the MC mechanism.

$T_1(b)$ containing the root.

Once $F(b)$ has been computed, for each agent $i \in R(b)$, define $Y(i, b)$ to be the node at or above i with minimum worth. The payment $P_i(b)$ of each agent i is then defined as

$$\begin{aligned} P_i(b) = &\quad \max(0, b_i - W_b(Y(i,b))) \\ = &\quad b_i - \min(b_i, W_b(Y(i,b))) \qquad \forall i \in R(b) \\ P_i(b) = &\quad 0 \qquad\qquad\qquad\qquad \forall i \notin R(v). \end{aligned} \tag{6.1}$$

We claim that this formula for the payment is equivalent to $P_i(b) = b_i q_i(b) - (W_b - W_b^{-i})$. To see this, we break into cases. If $q_i(b) = 0$ so that i does not receive the transmission, then removing i from the system does not change the overall net worth, since the net worth of $Y(i, b)$ is already negative, so it does not affect any of the nodes above it. Now suppose that $q_i = 1$, so i does receive the transmission. Then as i lowers its bid, the values $W_b(\beta)$ decrease at the same rate, for all nodes β at or above agent i. In particular, the node $Y(i, b)$ remains the same. This persists

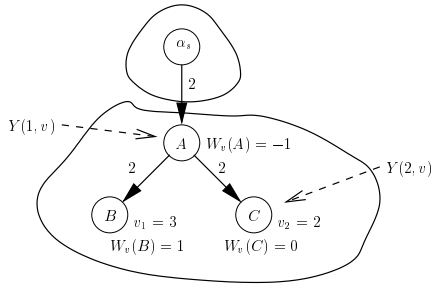

Figure 6.3: An opportunity for a successful pair strategy. The circled groups of nodes represent the two components of $F(v)$.

until either $b_i = 0$ or $W_b(Y(i,b))$, whichever occurs first. If it is the latter, then lowering b_i further does not change W_b. Thus, $W_b - W_b^{-i} = \min(b_i, W_b(Y(i,b)))$.

This completes our description of the algorithm. If there are multiple nodes at or above i with the same worth, we choose $Y(i,v)$ to be the one among them nearest to i; this does not alter the payment, but it simplifies our later results on when a coalition can be successful. For the same reason, we define $Y(i,b)$ for $i \notin R(b)$ to be the closest node at or above i that has zero or negative worth. We will use this characterization of the receiver set and payments in terms of $F(v)$ and $Y(i,v)$ in our analysis of group strategies against the MC mechanism.

Before launching into the analysis of successful group strategies, we now give two specific instances which will serve as canonical examples of such strategies.

Example 1 *Consider Figure 6.3. Here* $R(v) = \emptyset$*, so when both agents report truthfully, both attain a utility of zero. But if both lie by reporting* $b_1 = b_2 = 4$*, then* $R(b) = \{1, 2\}$ *and each pays* 2*, so agent* 2*'s utility remains at zero while agent* 1*'s rises to* 1*. Notice that* $b' = (3, 4)$ *would not be a successful group strategy, because in that case agent* 2 *would have to pay* 3*.*

Example 2 *Consider Figure 6.4. In this case, there is no successful pair strategy.*

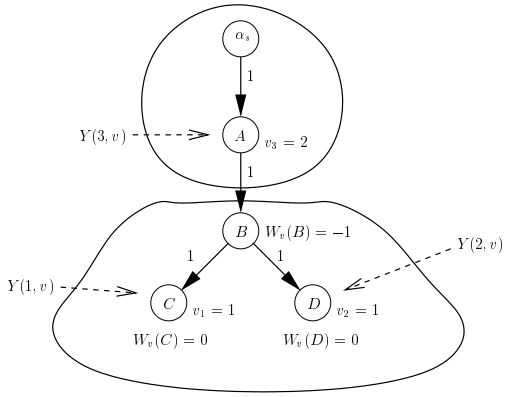

Figure 6.4: An opportunity for a basic triple strategy. The circled groups of nodes represent the two components of $F(v)$.

When all agents report truthfully, only agent 3 *receives the transmission, paying* $P_3(v) = 1$. *Agents* 1 *and* 2 *can never attain positive utility, because whenever either one appears in the receiver set, she must at least pay for the link directly above her. If only one of these two agents lies and she manages to join the receiver set, then she will effectively pay* 2 *for the two links above her. However, if agents* 1 *and* 2 *both lie so that the reported valuations are* $b = (2, 2, 2)$, *then* $R(b) = \{1, 2, 3\}$, *and* $P(b) = (1, 1, 0)$, *so agent* 3*'s utility rises from* 1 *to* 2 *and the other two agents remain at zero utility.*

To better understand these examples, it helps to interpret the payment formula (6.1) in a different way, in terms of *cutoff bids*.

Definition 6.3.1 *Fixing* b_{-i}, *the vector of reported utilities of all agents aside from* i, *suppose there is some number* $C_i(b_{-i})$ *such that if* $b_i < C_i(b_{-i})$ *then* $i \notin R(b)$, *and if* $b_i \geq C_i(b_{-i})$ *then* $i \in R(b)$ *and* i *pays* $P_i(b) = C_i(b_{-i})$. *Then we call* $C_i(b_{-i})$ *the* cutoff bid *for* i.

Lemma 6.3.2 *The cutoff bid always exists, and can be computed from* $W_b(\cdot)$ *as*

follows. For $i \in R(b)$, $C_i(b_{-i}) = \max\{0, b_i - W_b(Y(i,b))\}$. *For* $i \notin R(b)$, $C_i(b_{-i}) = b_i + L$, *where*

$$L = \sum_{\alpha \in N_b(i)} |W_b(\alpha)|$$

and $N_b(i)$ *denotes the set of nodes* α *at or above* i *such that* $W_b(\alpha) < 0$.

Proof: We consider what happens if i raises or lowers her reported valuation from b_i, while the other agents hold theirs fixed at b_{-i}.

If $i \in R(b)$, then all of the nodes at or above i have non-negative worth, and

$$P_i(b) = b_i - W_b(Y(i,b)), \tag{6.2}$$

provided $W_b(Y(i,b)) \leq b_i$. Since $Y(i,b)$ is the node of minimum worth at or above i, agent i can lower her reported valuation by $W_b(Y(i,b))$ before the worth of this node drops below zero, causing i to leave the receiver set. As b_i varies anywhere above this threshold, both terms on the right side of (6.2) change by the same amount, so i's payment remains unchanged. Thus, $C_i(b_{-i}) = b_i - W_b(Y(i,v))$. If $W_b(Y(i,b)) > b_i$, then i is in the receiver set and pays zero no matter what her reported valuation, so $C_i(b_{-i}) = 0$.

If $i \notin R(b)$ then there is some sequence of nodes $\alpha_1, \ldots, \alpha_k$ ($k \geq 1$) in this order along the path from i to α_s such that $W_b(\alpha_j) < 0$ for $j = 1, \ldots, k$. As i increases her reported valuation from b_i to $b_i + |W_b(\alpha_1)|$, the worth of each node from i to α_1 also increases by $|W_b(\alpha_1)|$. As i's reported valuation rises another $|W_b(\alpha_2)|$, all the nodes from i to α_2 increase their worth by the same amount. Inductively, we see that i first joins the receiver set when her reported valuation reaches $b_i + L$. As the reported valuation increases further, all nodes at or above i increase in worth. Since $W_{(b_{-i}, b_i+L)}(\alpha_k) = 0$, i's payment is $b_i + L$ when she first joins the receiver set. Further raising her reported valuation does not change her payment. ■

Armed with this new understanding of the payments, it is easy to detect when agent j has the opportunity to lie to increase the utility of another agent i. If $i \in R(v)$, then i's cutoff bid is already at most v_i, so j needs to somehow lower it further. If $i \notin R(v)$, then j needs to somehow lower i's cutoff bid below v_i, so that i joins the receiver set and attains positive utility. In both cases, this is possible if and only if $C_i(v_{-i}) > 0$ and j resides at or below node $Y(i, v)$, by Lemma 6.3.2. The tricky part is to figure out how j can do this without hurting herself – if j raises b_j enough to help i, then j will necessarily join the receiver set. (This is because b_j affects the worth of $Y(i, v)$ only if all nodes on the path from j to $Y(i, v)$ have strictly positive worth, and since i must be in the receiver set to have positive utility, all nodes above $Y(i, v)$ must have non-negative utility.) If $j \notin R(v)$, then $C_j(v_{-j}) > v_j$ so if the other agents continue to report truthfully, then j will be charged more than v_j, incurring negative utility. The pair strategy in Example 1 succeeds because A is the first negative worth node above agent 2, and agent 1 resides below A, so agent 1 can "protect" agent 2. The following definition and theorem formalize this observation.

Definition 6.3.3 *Define* $\mathcal{N}'(v) = \{i \in P | u_i(v) < v_i\}$. *Then* v *is said to admit a* pair opportunity *if there are agents* i *and* j *in the same component of* $F(v)$ *such that* $i \in \mathcal{N}'(v)$ *and* j *resides at or below* $Y(i, v)$.

Theorem 6.3.4 *There exists a successful pair strategy for agents* i *and* j *at valuation profile* v *if and only if* v *admits a pair opportunity for* i *and* j. *In this case, let*

$$L = \sum_{\alpha \in N_v(i)} |W_v(\alpha)|,$$

where $N_v(i)$ *is defined as in Lemma 6.3.2, and* $L' > L$. *Then* $v|^{\{i,j\}}(v_i + L, v_j + L')$

is a successful pair strategy for i and j.

Proof: *If direction:* Suppose i and j are both in $R(v)$. Then j can raise her reported valuation b_j without hurting herself, since v_j is already at least as large as j's cutoff bid. Doing so will increase the worth of all nodes above j, including the entire path from $Y(i, v)$ to α_s. This will decrease the price that i pays, since $Y(i, v)$ is the bottleneck node determining that price.

Now suppose that i and j are in some other component of $F(v)$. We have $L > 0$, since otherwise $i \in R(v)$. If j raises her reported valuation b_j to $v_j + L$, then i's cutoff bid will be exactly v_i. Since j resides at or below $Y(i, v)$, further raising j's reported valuation to $v_j + L'$ will make the worth of all nodes at or above i strictly positive, driving i's cutoff strictly below v_i. Since i and j are in the same component of $F(v)$, the root of this component is the first negative-worth node above each one (with respect to v). Therefore, if i also raises her reported valuation to $v_i + L$, then j's cutoff will be exactly v_j. Thus, by colluding in this way, i and j will both be included in the receiver set, j's utility will remain zero, and i's utility will increase. (Note: if i does not reside below $Y(j, v)$, then it is impossible for i to lower j's cutoff bid any lower than v_j.)

Only if direction: If i and j have a successful pair strategy, then the collusion must cause one of them to improve her utility, so without loss of generality we can assume it is i, hence $i \in \mathcal{N}'(v)$. We will argue using i and j's cutoff bids. Let $\hat{b}$ denote the successful pair strategy. Since all agents k aside from i and j have $\hat{b}_k = v_k$, we can consider i's cutoff $C_i(\hat{b}_{-i})$ to depend only on $\hat{b}_j$, and similarly j's cutoff bid to depend only on $\hat{b}_i$.

If $i \in R(v)$, then the only way to improve i's utility is to lower her cutoff bid, which j can do only if she resides at or below $Y(i, v)$. If $i \notin R(v)$, then i's cutoff

is initially above v_i and must be lowered strictly below v_i.

In the latter case, we know that $W_v(Y(i,v)) \leq 0$, and there is some node of strictly negative worth above i. Since the strategy $\hat{b}$ improves i's utility, we know by Lemma 6.3.2 that $W_{u|^j\hat{b}_j}(\alpha) > 0$ for all nodes α at or above i. In order for j to cause i to connect and pay less than v_i, $\hat{b}_j$ must be high enough to make all of the nodes at and above $Y(i,v)$ have strictly positive worth with respect to $v|^j\hat{b}_j$. In particular, j must reside at or below $Y(i,v)$, since otherwise b_j does not affect the worth of $Y(i,v)$. Moreover, all nodes at or above j must have strictly positive worth with respect to $v|^j\hat{b}_j$. If $\hat{b}_i \geq v_i$, then these nodes also have strictly positive worth with respect to $\hat{b}$. If $\hat{b}_i < v_i$, then $W_{\hat{b}}(\alpha)$ may be lower than $W_{u|^j\hat{b}_j}(\alpha)$ for some of the nodes α at or above i, but all of these still must have non-negative worth, since otherwise i would not be in the receiver set, so would not attain positive utility. Therefore, j is also in the receiver set $R(\hat{b})$.

Suppose i and j reside in different components of $F(v)$. Since $Y(i,v)$ is in i's component and j resides below this node, we know j's component lies below i's. Let α denote the root of j's component in $F(v)$. We just argued that $j \in R(\hat{b})$. But i's reported valuation has no effect on $W_b(\alpha)$, which is negative when $b = v$ (i.e., when all utilities are reported truthfully). Thus, j's cutoff $C_j(v_{-j}|^i\hat{b}_i)$ is at least $v_j + |W_v(\alpha)|$, so $u_j(\hat{b}) < 0$, which contradicts the assumption that $\hat{b}$ is a successful strategy. Thus, i and j must reside in the same component of $F(v)$. ■

In Example 2, v admits no pair opportunity, so by Theorem 6.3.4, there is no successful pair strategy. Yet, the three agents can still collude successfully. This is because agents 1 and 2 are in the same component of $F(v)$, hence can protect each other from incurring negative utility, and reside below $Y(3,v)$, hence can help agent 3. It turns out that this situation is the only other way in which a successful

group strategy can arise. We now formalize this result.

Definition 6.3.5 *Suppose $i \in \mathcal{N}'(v)$ and there is a component of $F(v)$, distinct from i's, that lies below $Y(i,v)$ and contains agents j and k. Then we say that v presents a* triple opportunity *for $\{i,j,k\}$.*

Definition 6.3.6 *Suppose v presents a triple opportunity for $\{i,j,k\}$. Let*

$$L = \sum_{\alpha \in N_v(j)} |W_v(\alpha)|$$

and $L' > L$. Then the strategy $b_{\{i,j,k\}} = (v_i, v_j + L', v_k + L')$ is called a basic triple strategy *for $\{i,j,k\}$.*

Note that if either j resides at or below $Y(k,v)$ and $k \in \mathcal{N}'(v)$ or vice versa, then j and k have a successful pair strategy, by Theorem 6.3.4. But even if neither of those conditions holds, the next claim shows that the basic triple strategy still succeeds.

Claim 6.3.7 *Suppose v presents a triple opportunity for $\{i,j,k\}$. Then the corresponding basic triple strategies are successful group strategies for $\{i,j,k\}$.*

Proof: By assumption, $j,k \notin R(v)$, so these agents intially have zero utility. Since j and k are in the same component of $F(v)$, we have $N_v(j) = N_v(k)$, the lowest node in this set being the root of this component. Thus, $W_{u|^j(v_j+L')}(\alpha) > 0$ for all $\alpha \in N_v(k)$, so $C_k(v_{-k}|^j(v_j + L')) \leq v_k$. Similarly, $C_j(v_{-j}|^k(v_k + L')) \leq v_j$. Thus, under the group strategy $\hat{b} = u|^{\{j,k\}}(v_j+L', v_k+L')$, j and k at least maintain their zero utility.

Since $N_v(i) \subset N_v(j)$, $L' > L$, and j resides below $Y(i,v)$, we have

$$W_{u|^j(v_j+L')}(\alpha) > 0$$

for all α at or above i. Increasing b_k as well only increases the worths, so $i \in R(\hat{b})$ and $P_i(\hat{b}) < v_i$, so $u_i(\hat{b}) > 0$. Thus, if $i \notin R(v)$, the strategy improved i's utility. If $i \in R(v)$, the strategy improves i's utility because $L' > L$ guarantees $W_{\hat{b}}(Y(i,v)) > W_v(Y(i,v))$, so j and k's increased reported utilities lowered i's cutoff bid. ■

Definition 6.3.8 *If I is a set of agents, we say that $i \in I$ is* minimal with respect to $F(v)$ *if there is no other agent $j \in I$ located in a different component of $F(v)$ such that i's component of $F(v)$ contains any nodes above j. That is, if we contract components of $F(v)$, there is no agent $j \in I$ who resides strictly below i.*

Theorem 6.3.9 *If some coalition S has a successful group strategy at v, then either there exist agents $i, j \in S$ with a successful pair strategy, or there exist agents $i, j, k \in S$ with a successful basic triple strategy. Conversely, if there is a pair or triple of agents with a successful pair or basic triple strategy, then every set of agents S containing that pair or triple has a successful group strategy.*

Proof: Denote the group strategy by $\hat{b}$. Since the strategy succeeds, the set $I = \{i \in S : u_i(\hat{b}) > u_i(v)\}$ is non-empty. Select some agent $i \in I$ that is minimal with respect to $F(v)$. Since i benefitted from the strategy $\hat{b}$ as compared to v, the manipulations of the other agents in S must have reduced i's cutoff bid, and $i \in R(\hat{b})$. Thus S must include some other agent h residing at or below $Y(i,v)$, such that $\hat{b}_h > v_h$ and the worth $W_{\hat{b}}(\alpha)$ of each node α between h and $Y(i,v)$ is strictly positive. Let J denote the set of all such agents in S, and select some agent $j \in J$ that is minimal with respect to $F(v)$. Note that $j \in R(\hat{b})$, since (under $\hat{b}$) it has a positive worth path to $Y(i,v)$, and $i \in R(\hat{b})$. If j lies in the same component of $F(v)$ as i, then i and j have a successful pair strategy, by Theorem 6.3.4. Otherwise, j's component lies below i's.

Suppose j's component contains no other agents in S, and let β denote the root

of that component. Because j is minimal in J, even if there is some agent $k \in S$ who resides in a component strictly below j's, then her misreported valuation has no positive effect on the net worths computed at the nodes between j and β. This is because $j \in J$ but $k \notin J$, so there is some node α at or above k but not at or above j such that $W_{\hat{b}}(k) \leq 0$. Thus, j's cutoff bid is at least $v_j + |W_v(\beta)| > v_j$. So $u_j(\hat{b}) < 0$, which contradicts the assumption that $\hat{b}$ is a successful strategy. Thus, j's component of $F(v)$ contains some other agent $k \in S$. Thus, by Claim 6.3.7, there is a successful basic triple strategy available for $\{i, j, k\}$.

The converse holds because the successful pair and basic triple strategies involve raising reported utilities, which can never increase the cutoffs for the other agents. So if each other agent h reports her true valuation v_h, then her utility does not decrease, so she can be considered to be part of the strategizing set. ■

Summarizing the previous results, we have the following characterization of valuation profiles for which the MC mechanism is GSP.

Theorem 6.3.10 *The MC mechanism is GSP at v if and only if the following condition holds for each $i \in \mathcal{N}'(v)$: there is no agent j in the same component of $F(v)$ as i such that j resides at or below $Y(i, v)$, and there is no pair of agents j and k residing in the same component as each other and below $Y(i, v)$.*

We have now completely characterized the valuation profiles v at which the MC mechanism is GSP, shown that the minimal sets of agents who can successfully manipulate the mechanism are pairs and triples, and shown that every set of agents containing such a pair or triple has a successful group strategy. The question remains, what do successful group strategies look like in general? The following theorem shows that every group strategy can be converted into a "canonical" one.

Theorem 6.3.11 *If S has a successful strategy $\hat{b}$ at true valuation profile v, then $S' = S \cap R(\hat{b})$ has a successful strategy $\hat{b}'$ such that*

- $S' \subseteq R(\hat{b}')$
- $u_i(\hat{b}') \geq u_i(\hat{b})$ *for all* $i \in \mathcal{N}$
- $\hat{b}'_i \geq v_i$ *for all* $i \in S'$

Proof: For each $i \in S - R(\hat{b})$ such that $\hat{b}_i > v_i$, set $\hat{b}'_i = v_i$. This may raise the cutoff bids for some agents outside $R(\hat{b})$, but these had zero utility anyway. It has no effect on the computed net worths for nodes in the root component of $F(\hat{b})$, hence no effect on the receiver set or on the price charged to any node in $R(\hat{b})$. Now set $\hat{b}'_i = v_i$ for each remaining $i \in S - R(\hat{b})$. This can only decrease the prices paid by agents in $R(\hat{b})$. It may also expand the receiver set, but that will not cause any of the new recipients to get negative utility because we have already gotten rid of all the agents in $S - R(\hat{b})$ who exaggerated their utilities. Now set $\hat{b}'_i = v_i$ for each $i \in S \cap R(\hat{b})$ such that $\hat{b}_i < v_i$. This does not change the receiver set, and can only decrease the prices paid by the receivers. Since S had some agent who benefitted from the strategy and this agent is in $R(\hat{b})$, we can now throw all agents in $S - R(\hat{b})$ out of the strategizing set, leaving us with S'. ■

Thus, in some sense, the "interesting" group strategies are the ones in which the agents who misreport their utilities only exaggerate them, and all of them end up in the receiver set.

BIBLIOGRAPHY

[1] M. Adler and D. Rubenstein. Pricing multicast in more practical network models. In *Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithms*, pages 981–990, 2002.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows : Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[3] A. Archer, J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker. Approximation and collusion in multicast cost sharing. *Games and Economic Behavior*. To appear.

[4] A. Archer, C. Papadimitriou, K. Talwar, and É. Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 205–214, 2003.

[5] A. Archer and É. Tardos. Truthful mechanisms for one-parameter agents. In *Proceedings of the* 42^{nd} *IEEE Symposium on Foundations of Computer Science*, pages 482–491, 2001.

[6] A. Archer and É. Tardos. Frugal path mechanisms. In *Proceedings of the* 13^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 991–999, 2002.

[7] K. J. Arrow. *Social Choice and Individual Values*. Wiley, New York, 1951.

[8] A. Ballardie, P. Francis, and J. Crowcroft. Core based trees. In *Proceedings of ACM SIGCOMM '93*, pages 85–95, 1993.

[9] Z. Bar-Yossef, K. Hildrum, and F. Wu. Incentive-compatible online auctions for digital goods. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 964–970, 2002.

[10] D. Baron and R. Myerson. Regulating a monopolist with unknown costs. *Econometrica*, 50:911–930, 1982.

[11] S. Bikhchandani, S. de Vries, J. Schummer, and R. V. Vohra. Linear programming and Vickrey auctions. In B. Dietrich and R. V. Vohra, editors, *Mathematics of the Internet: E-Auction and Markets*, IMA Volumes in Mathematics and its Applications, Volume 127, pages 75–116. Springer, New York, 2002.

[12] V. Chari and R. Weber. How the U.S. Treasury should auction its debt. *Federal Reserve Bank of Minneapolis Quarterly Review*, 16(4):3–12, 1992.

[13] C. Chekuri and S. Khanna. A PTAS for minimizing weighted completion time on uniformly related machines. In F. Orejas, P. G. Spirakis, and J. van Leeuwen, editors, 28^{th} *International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science, Volume 2076, pages 848–861. Springer, Berlin, 2001.

[14] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 8:17–33, 1971.

[15] R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling*. Addison-Wesley, Reading, MA, 1967.

[16] P. Cramton. Spectrum auctions. In M. Cave, S. Majumdar, and I. Vogelsang, editors, *Handbook of Telecommunications Economics*, pages 605–639. Elsevier, Amsterdam, 2002.

[17] S. Deering and D. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, 8:85–110, 1990.

[18] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. The PIM architecture for wide-area multicast routing. *ACM-IEEE Transactions on Networking*, 4:153–162, 1996.

[19] E. Elkind, A. Sahai, and K. Steiglitz. Frugality in path auctions. Unpublished manuscript, 2003.

[20] J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker. Hardness results for multicast cost sharing. *Theoretical Computer Science*, 304:215–236, 2003.

[21] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. In *Proceedings of the 2002 ACM Symposium on Principles of Distributed Computing*, pages 173–182, 2002.

[22] J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63:21–41, 2001.

[23] A. Fiat, A. Goldberg, J. Hartline, and A. Karlin. Competitive generalized auctions. In *Proceedings of the* 34^{th} *ACM Symposium on Theory of Computing*, pages 72–81, 2002.

[24] M. Friedman. How to sell government securities. *Wall Street Journal*, page A8, August 28, 1991.

[25] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, Cambridge, 1991.

[26] A. Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41:587–601, 1973.

[27] M. X. Goemans and M. Skutella. Cooperative facility location games. In *Proceedings of the* 11^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 76–85, 2000.

[28] A. Goldberg, J. Hartline, and A. Wright. Competitive auctions and digital goods. In *Proceedings of the* 12^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 735–744, 2001.

[29] T. Gonzalez and S. Sahni. Preemptive scheduling of uniform processor systems. *Journal of the ACM*, 25:92–101, 1978.

[30] J. R. Green and J.-J. Laffont. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica*, 45:427–438, 1977.

[31] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[32] R. Guesnerie and J.-J. Laffont. A complete solution to a class of principal-agent problems with an application to the control of a self-managed firm. *Journal of Public Economics*, 25:329–369, 1984.

[33] J. Hershberger and S. Suri. Vickrey prices and shortest paths: What is an edge worth? In *Proceedings of the* 42^{nd} *IEEE Symposium on Foundations of Computer Science*, pages 252–259, 2001.

[34] S. Herzog, S. Shenker, and D. Estrin. Sharing the 'cost' of multicast trees: An axiomatic analysis. *ACM/IEEE Transactions on Networking*, 5:847–860, 1997.

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

[36] H. Holbrook and D. Cheriton. IP multicast channels: EXPRESS support for large-scale single-source applications. In *Proceedings of ACM SIGCOMM '99*, pages 65–78, 1999.

[37] E. Horvath, S. Lam, and R. Sethi. A level algorithm for preemptive scheduling. *Journal of the ACM*, 24:32–43, 1977.

[38] K. Jain and V. V. Vazirani. Applications of approximation algorithms to cooperative games. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 364–372, 2001.

[39] V. Krishna. *Auction Theory*. Academic Press, New York, 2002.

[40] R. Lavi and N. Nisan. Competitive analysis of incentive compatible on-line auctions. In *Proceedings of the* 2^{nd} *ACM Conference on Electronic Commerce*, pages 233–241, 2000.

[41] D. Lehmann, L. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49:577–602, 2002.

[42] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.

[43] P. Malvey and C. Archibald. Uniform-price auctions: Update of the Treasury experience. Technical Report 3103, U.S. Treasury Department Office of Public Affairs, Washington, D.C., October 1998.

[44] K. R. Mandelkow, editor. *Goethes Briefe*. Wegner, Hamburg, 1968.

[45] A. Mas-Collel, M. Whinston, and J. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[46] E. Maskin and J. Riley. Monopoly with incomplete information. *Rand Journal of Economics*, 15:171–196, 1984.

[47] R. P. McAfee and J. McMillan. Multidimensional incentive compatibility and mechanism design. *Journal of Economic Theory*, 46:335–354, 1988.

[48] A. Meyerson. Online facility location. In *Proceedings of the* 42^{nd} *IEEE Symposium on Foundations of Computer Science*, 2001.

[49] J. Mirrlees. An exploration in the theory of optimum income taxation. *Review of Economic Studies*, 38:175–208, 1971.

[50] B. Moldovanu and M. Tietzel. Goethe's second-price auction. *Journal of Political Economy*, 106:854–859, 1998.

[51] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, 1995.

[52] H. Moulin. Incremental cost sharing: characterization by strategyproofness. *Social Choice and Welfare*, 16:279–320, 1999.

[53] H. Moulin and S. Shenker. Strategyproof sharing of submodular costs: Budget balance versus efficiency. *Economic Theory*, 18:511–533, 2001.

[54] A. Mu'alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *Proceedings of the* 18^{th} *National Conference on Artificial Intelligence and* 14^{th} *Conference on Innovative Applications of Artificial Intelligence*, pages 379–384, 2002.

[55] M. Mussa and S. Rosen. Monopoly and product quality. *Journal of Economic Theory*, 18:301–317, 1978.

[56] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proceedings of the* 31^{st} *Annual ACM Symposium on Theory of Computing*, pages 129–140, 1999.

[57] N. Nisan and A. Ronen. Computationally feasible VCG mechanisms. In *Proceedings of the* 2^{nd} *ACM Conference on Electronic Commerce*, pages 242–252, 2000.

[58] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[59] M. J. Osborne and A. Rubinstein. *A Course in Game Theory.* MIT Press, Cambridge, MA, 1994.

[60] R. Perlman, C.-Y. Lee, A. Ballardie, J. Crowcroft, Z. Wang, T. Maufer, C. Diot, and M. Green. Simple multicast: A design for simple low-overhead multicast. Technical report, IETF Internet Draft, 1999.

[61] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37:130–143, 1988.

[62] P. Raghavan and C. D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.

[63] K. Reynolds. Government securities – the scandal of '91, 1996. Available from http://www.agorics.com/Library/Auctions/auction10.html.

[64] K. Roberts. The characterization of implementable choice rules. In J.-J. Laffont, editor, *Aggregation and Revelation of Preferences*, pages 321–348. North-Holland, Amsterdam, 1979.

[65] M. A. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.

[66] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, New York, 1986.

[67] K. Talwar. The price of truth: Frugality in truthful mechanisms. In 20^{th} *Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, Volume 2607, pages 608–619. Springer, Heidelberg, 2003.

[68] V. Vazirani. *Approximation Algorithms.* Springer-Verlag, New York, 2001.

[69] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.