A Modular Framework for Multi-Agent Preference Elicitation

A dissertation presented

by

Sébastien Lahaie

to

The School of Engineering and Applied Sciences

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the subject of

Computer Science

Harvard University

Cambridge, Massachusetts

October 2007

Thesis advisor

David C. Parkes

Author

Sébastien Lahaie

A Modular Framework for Multi-Agent Preference Elicitation

Abstract

I present a framework for multi-agent preference elicitation in the context of a discrete resource-allocation problem, known as the *combinatorial allocation problem* (CAP). There are several distinct, indivisible items, which must be allocated among a set of agents. The agents value bundles rather than just individual items. Because the number of bundles can be very large, agent preferences cannot be exhaustively described. An elicitation scheme for the CAP must therefore carefully choose the language in which it will model agent preferences to ensure succinct representations.

The approach I propose is to embed learning algorithms for certain preference representations into the resource-allocation process. Preferences are elicited incrementally, and at well-defined breakpoints a tentative allocation is computed. This process is repeated to the extent needed until an efficient allocation is found. The framework is modular in that a variety of different learning algorithms can be introduced as subroutines to construct models of the individuals agents' preferences, as long as the subroutines interact with the agents through a standard query interface.

The current leading distributed algorithms for the CAP are iterative combinatorial auctions, but the iterative combinatorial auctions that can guarantee allocative efficiency all use a single bidding language, namely XOR, that may not be appro-

priate for certain applications. Experimental results demonstrate that the elicitation framework can complement current designs by allowing for alternate representations where XOR is inappropriate, resulting in fewer queries and faster convergence.

The framework consists of elicitation, allocation, and pricing engines. The pricing engine is also modular. I present two different methods for pricing, one of which can also serve as a stand-alone iterative auction. The auction begins with item prices and introduces bundle prices as needed to drive the bidding forward. This again complements existing designs which are limited to XOR or item pricing. The framework can also be extended to compute VCG payments, to bring truthful responses to queries into an equilibrium.

Contents

Acknowledgments

I thank my advisor, David Parkes, for his constant support and guidance throughout my graduate career. I have learned more than I could ever have expected through his example, and this dissertation attests to his influence on my development as a researcher.

I thank Dave Pennock for serving as a mentor during my summers at Yahoo, and for introducing me to a wealth of fascinating and important research problems. It will be a pleasure to keep working alongside him in the coming years.

I thank Jay Aslam for introducing me to research and computational learning theory while I was an undergraduate. His early influence is also clearly reflected in this thesis.

I thank the many friends and colleagues who made graduate school such an enjoyable and stimulating experience, especially Giro Cavallo, Florin Constantin, Jacomo Corbo, Kobi Gal, Geoff Goodell, Shaili Jain, Laura Kang, Adam Kirsch, Konrad Lorincz, Loizos Michael, Jill Nickerson, Tim Rauenbusch, and Pedro Sander.

I thank my parents and my extended family. Without their unconditional support this thesis would never have been possible.

Finally, I thank my fiancée Eleni for her encouragement and support, especially during the difficult final months of this thesis. She kept me afloat during the long hours of coding and writing with her perspective, patience, and humor. From the moment I knew her, things somehow started to quickly fall into place: conjectures were resolved, experiments were completed, job offers came in, and the thesis was written. She has given me focus and purpose.

Chapter 1

Introduction

Open networks such as the Internet are today fundamentally changing the nature of business-to-consumer and business-to-business interactions. Consumers now have unprecedented access to a wealth of information on products and services with simple Web browsing, and the available information keeps growing. Businesses, on the other hand, can now keep careful track of the behavior of online consumers, opening up possibilities for much more powerful targeting and product placement. With this proliferation of information comes added complexity, and hence the need for efficient methods to organize and distill information into a useful format, so that users can act on it.

Users are increasingly turning to the online world not just to push and pull information, but to also interact with one another in social and commercial environments. People from all walks of life now keep up with their contacts through social networks such as Facebook, LinkedIn, and MySpace. Blogs allow the dissemination and discussion of a wide variety of opinions. Auction sites such as eBay allow individuals and

small businesses to engage in commerce with each other online. Search portals such as Google and Yahoo let advertisers promote their products and services next to search query results. Besides grappling with the complexity of organizing information, the design of electronic services must then also be infused with insights into the social and commercial implications of various design decisions.

These considerations point to the need for formal iteraction protocols that respect the information-processing limitations of end users, as well as the principles of trust, reputation, privacy, and honesty that lay the foundations for productive social and commercial engagements.

1.1 Preference Elicitation

Preference elicitation is the process of creating a model of a user's preferences to the extent needed to make choices based on these preferences. In a single-agent setting, preference elicitation schemes are typically used for decision support; examples include travel planning [65], product recommendation [21], and sales assistance [103]. Due to cognitive limitations, a user may not be able to identify the correct choice to make with respect to his preferences in a given setting. A decision support system queries the user through a formal interface, in a systematic fashion, in order to recover a model of his preferences [23]. The decision-making can then be automated on behalf of the user. The model must satisfy two requirements to be effective: it should closely and succinctly approximate the user's preferences, and should allow for fast computation of the optimal decision. The choice of preference model is therefore central to the design of a decision support system.

In multi-agent preference elicitation, the aim is analogous. There are several agents, and we wish to recover a model of their preferences in order to implement an optimal system-wide outcome, where the criterion used to evaluate solutions is context-dependent. Again, the focus is on representation: to ensure an effective working protocol, the preference models must be carefully chosen to succinctly approximate agent preferences, and should seamlessly integrate into an algorithm for computing system-wide solutions.

The multi-agent setting introduces several new challenges. Rather than just recovering models of the individual preferences independently, a sophisticated multi-agent protocol can and should use its knowledge of one agent's preferences to guide the elicitation of another's. Given multiple preference models, it is also often more difficult to compute a system-wide solution compared to the scenario with a single agent. Finally, whereas with a single agent the elicitation scheme is usually devoted to this agent's satisfaction, a multi-agent system must contend with multiple and typically conflicting interests when implementing a system-wide solution. As a result, there is the distinct and real possibility that agents may try adapt their input in order to achieve a favorable outcome for themselves. The incentives of the participants must therefore be taken into account in the protocol design.

To design an effective multi-agent preference elicitation scheme, we need to draw on techniques from several disciplines. Structured preference models fall within the domain of artificial intelligence [91]; examples from this literature include conditional preference networks [18, 19] and logical bidding languages [20] (designed expressly for multi-agent settings), to name a few. Algorithms for computing system-wide out-

comes draw on ideas from operations research, particularly integer programming and combinatorial optimization [73], because of the wide applicability of these techniques. Incentives in multi-agent environments are the purview of mechanism design, a subfield of microeconomics [67]. Mechanism design provides precise characterizations of the outcomes that can be implemented when agents are self-interested.

In this dissertation, I present a framework for multi-agent preference elicitation in the context of a discrete resource-allocation problem, called the *combinatorial allocation problem* (CAP). There are several distinct, indivisible goods, which must be allocated among a set of agents. The agents' preferences are combinatorial: certain goods may be superfluous in the presence of others (substitutes), while some goods may be worth more together than individually (complements). This is a canonical resource allocation problem, with application in domains such as bandwidth allocation [38], airport scheduling [88], and vehicle routing [61]. The framework I propose is modular in that a variety of different subroutines can be introduced to construct models of the individuals agents' preferences, as long as the subroutines interact with the agents through a standard query interface.

1.2 Combinatorial Auctions

Auctions are arguably the most classic distributed resource allocation mechanisms [48]. An auction can be construed as a formalized market mechanism that elicits value information from agents in the form of bids, and decides on an allocation accordingly. As such, an iterative auction is a preference elicitation protocol. An auction that allocates several distinct indivisible items is commonly called a "com-

binatorial" or "package" auction. The field of combinatorial auctions has burgeoned in recent years, due to important application domains in both the public and private sectors, such as spectrum allocation [68] and supply-chain management [108]. A recent textbook edited by Cramton et al. [28] provides a compilation of research in this area.

Today's leading iterative combinatorial auction designs represent the state-of-the-art in multi-agent preference elicitation for the combinatorial allocation problem, and are the benchmark against which I will evaluate my own preference elicitation scheme. Indeed, the preference elicitation framework given in this dissertation can be also interpreted as an auction—or more precisely, an auction skeleton—in the sense that it uses prices to provide agents with feedback and clear the market.

1.2.1 Single-Shot Designs

Before turning to preference elicitation in the form of iterative auctions, it is instructive to first survey the current state of technology for single-shot auctions. The central component of today's single-shot combinatorial auctions is a *winner-determination* engine. Given the bidders stated preferences (in the form of bids), a typical winner-determination engine formulates the allocation as a mixed-integer program (MIP), and then solves the MIP using established integer programming techniques, and perhaps proprietary heuristics as well. Algorithms for the winner-determination problem have been proposed and studied extensively in the OR and AI literatures [93, 95, 97, 98].

Because winner-determination engines rely on MIP techniques, bidding languages

for single-shot combinatorial auctions are designed so that they have straightforward conversions to MIP formulations. Indeed, all the bidding languages proposed to date in the AI literature have this property. The XOR, OR, and OR* languages, as well as combinations of these, all describe preferences in terms of a set of primitive "atomic bids" (bids on single bundles), which correspond to MIP variables, together with exclusivity constraints on these primitives, which easily translate to MIP constraints [74]. Languages that use logical connectives to describe complements and substitutes have also been proposed [20, 22], and again naturally translate to MIP formulations. Because of the general flexibility of MIP formulations, a wealth of plausible bidding languages are available for single-shot CAs.

The fact that it is possible to express combinatorial preferences means that a single-shot CA can achieve greater allocative efficiency than if the goods were auctioned off individually. This accounts for the growing use of single-shot CAs in the private and public sectors. The most common application of such auctions is for procurement purposes. For example, bus routes in London are allocated using a CA [40]. Freight transportation services in the U.S. are commonly allocated via CA as well; a number of companies provide commercial software for this purpose [45].

These single-shot CAs do not perform any "preference elicitation," unless we broaden the concept so widely as to trivialize the problem. A single-shot CA "elicits" preferences in the sense that bids represent preferences, but does not interact with agents through a fixed set of primitive queries, and does not provide bidder feedback. In some instances, procurement CAs are iterated for a small number of stages, and bidders are provided with some kind of ad-hoc feedback such as the current winning

bids or bidders [10]. In these cases, the burden of determining bid values and structures at each stage still lies entirely with the bidders. Nevertheless, single-shot or rudimentary multi-stage CAs dominate the landscape, at least in the private sector. Bichler et al. [10, 94] report that in their discussions with CA technology vendors, none mentioned applications of clock auctions, the type of iterative auction that has shown the most promise in laboratory experiments [87]. The reason is that clock auctions provide the wrong kind of expressiveness: businesses prefer to specify side constraints and volume discounts rather than place large numbers of package bids [96].

1.2.2 Iterative Designs

In contrast to single-shot CAs, many iterative CAs can rightfully be termed preference elicitation mechanisms. The typical information required at each round of an iterative CA is the *best-response set*: the auctioneer quotes prices, and bidders report their most preferred bundles at these prices. A simple query interface means that an iterative CA design can have the potential to port to a variety of different application domains.

Pros and Cons

There are several pros and cons to iterative auctions, compared to single-shot alternatives. Cramton [27] provides a detailed comparison of the two paradigms. The main microeconomic tradeoffs are summarized here.

Efficiency. The price discovery process inherent in iterative auctions leads to high levels of efficiency [27]; it allows bidders to focus their efforts on refining those

parts of their preferences that are most relevant to determining an efficient allocation. While single-shot counterparts may be efficient in theory, it can be costly for bidders to refine every detail of their valuations, and in practice this can result in inefficient outcomes if bidders then resort to approximations [82].

Revenue. An open, iterative auction can increase competition as participants observe each others' bids and subsequently try to outbid each other, leading to higher revenues. Milgrom and Weber [70] lend theoretical rigor to this intuition in the case where bidders' values are affiliated. On the other hand, if there are strong asymmetries in bidder values, weaker bidders may be discouraged from participating in an iterative auction, since their chances of losing are virtually guaranteed [51]. This can then result in very low revenues.

Privacy. In a single-shot auction, bidders must reveal their entire preferences to the auctioneer. This raises privacy concerns, since preferences reveal a lot about a bidder's priorities or business plan. Rothkopf et al. [90] have suggested that privacy concerns are one reason for the rarity of the Vickrey auction in practice, as opposed to its iterative counterpart, the English auction. In constrast, iterative auctions only elicit partial preferences, and full preferences are revealed only in the worst case. Cryptographic techniques can introduce more privacy in single-shot CAs, but have not yet been applied to auctions where bidders have combinatorial preferences [85].

Transparency. In an open iterative auction, bidders can observe that the auction was properly conducted and that the correct outcome was indeed implemented.

This gives the process some legitimacy. This is particularly important in the public sector, where auctioneers are accountable to their respective government agencies and to the public. Indeed, many iterative designs not only converge to an efficient allocation, but also provide a proof of optimality in the form of market-clearing prices (we will see in Chapter 3 how prices can serve as a proof).

Collusion. A drawback of iterative auctions is the potential for collusive behavior. The bidding process can turn into a communication mechanism, which the bidders could use to coordinate and achieve lower prices. The potential for collusion also exists in single-shot auctions, but the absence of communication during the auction makes it harder to enforce collusion.

Taken together, these considerations show that iterative auctions can bring many benefits compared to their single-shot counterparts, although the designs should be carefully tailored to avoid drawbacks such as collusive behavior.

Current Designs

Table 1.1 lists several well-known iterative combinatorial auctions from the academic literature. The first three provide formal guarantees on the level of efficiency they generate. The *i*Bundle auction is due to Parkes [78]. This same auction with a direct-revelation proxy interface is the Ascending-proxy auction, due to Ausubel and Milgrom [7]. Both auctions can guarantee efficiency that is arbitratily close to optimal. The next auction listed is due to deVries et al. (dVSV) [31], and can achieve exact efficiency in a finite number of rounds.

Name	Bid structure	Price structure	Query interface	Outcome
*i*Bundle	XOR	non-anon XOR	demand	exact
Ascending-proxy	XOR	non-anon XOR	–	exact
dVSV	XOR	non-anon XOR	best-response	exact
RAD	OR	linear	–	approximate
AkBA	XOR	anon XOR	demand	approximate
ICE	TBBL	linear	–	exact

Table 1.1: Elicitation properties of some well-known combinatorial auctions.

The next two auctions do not provide formal guarantees, but have been shown to perform well in either laboratory experiments or computer simulations. The Resource Allocation Design (RAD), due to Kwasnica et al. [56], has performed well in experiments against the simultaneous ascending auction. The latter has been used to allocate wireless spectrum, although it is not a package auction. The Ascending k-Bundle Auction (AkBA) is due to Wurman and Wellman [110], and has the distinction that it uses anonymous bundle pricing (i.e. each bidder sees the same price for each bundle). The effectiveness of AkBA was validated through computer simulations.

The Iterative Combinatorial Exchange (ICE), due to Parkes et al. [83], can be specialized to function as an auction. It is the only instance that provides both formal guarantees of efficiency as well as a bidding language other than XOR; however, it retains some aspects of single-shot designs because bidders must provide a tree structure to their valuations in order to participate, rather than just iteratively bid on packages.

For each auction, Table 1.1 gives the representations used for bidding and pricing. It also specifies what query interface the auction implements, if any. The final column

specifies whether the auction can guarantee exact or only approximate efficiency. (If the auction can guarantee efficiency that is arbitrarily close to optimal, we term this exact.) The efficiency guarantees hold if agents bid truthfully in the auction; we postpone strategic considerations to Section 1.4 below.

There are several things to note about this list. First, given the variety of bidding languages available [20, 74], it is perhaps striking that the auctions are all restricted to the OR or XOR language—with the exception of ICE, which is a relative newcomer to the scene. The auctions that can guarantee arbitrarily high levels of efficiency all use the XOR language. This can be highly problematic if the bidders' preferences cannot be succinctly represented using XOR: Nisan and Segal [76] have shown that in the worst-case, an efficient protocol must recover at least one bidder's entire preferences (up to a constant).

The selection of pricing structures used in the auctions is even more limited: all use either an anonymous or non-anonymous version of XOR, or simple linear prices. Linear prices—where the price of a bundle is obtained by summing the prices of its constituent items—are simple for bidders to interpret but cannot guarantee efficiency that is arbitrarily close to optimal. The same holds for anonymous XOR prices. Meanwhile, Nisan and Segal [76] show that in any efficient protocol, prices must coincide with some bidder's preferences in the worst-case, so a non-anonymous XOR pricing scheme is prone to the same failures as its bidding language counterpart.

The auctions also vary in the burden they place on agents to encode their preferences. In keeping with the spirit of preference elicitation through simple, intuitive interfaces, it would be ideal if bidders only had to interact with the system through

simple queries. In a market environment, there are two natural types of queries: value and demand queries. On a value query, the bidder must evaluate its value for a single bundle. On a demand query, the bidder must identify the bundle it prefers the most given a set of prices.

Only *i*Bundle and A*k*BA use a basic demand query interface. The dVSV auction requires best-response sets at each round; this dissertation shows that best-response queries can be simulated with value and demand queries. The remaining auctions do not implement any clear, primitive query interface, and require bidders to perform some pre-processing to encode their preferences in a useful format, as in single-shot auctions. The Ascending-proxy auction requires bidders to reveal their preferences in an XOR format up front to the proxies. The ICE protocol requires bidders to provide a tree structure for their preferences in the TBBL [22] language up front, and only value information is refined during the auction. Similarly, bidders must give an OR structure to their preferences beforehand in order to participate in RAD.

1.3 The Framework

The preference elicitation scheme I propose can in principle work for a variety of bidding languages, and it can be calibrated to achieve arbitrarily high levels of efficiency. As such, it represents a step towards the development of iterative auctions that could allow for representations tailored to specific application domains.

The key is to provide a learning algorithm with value and demand queries for the bidding language of choice. Learning algorithms exist for the XOR and OR languages, and for the language of Polynomials, whose application to elicitation is

novel to this work. The idea of embedding learning algorithms as subroutines for preference elicitation was given by Lahaie and Parkes [59]. Zinkevich et al. [111] were the first to draw connections between learning and elicitation, but they use a model that is restricted to linear prices. As a result, they cannot guarantee efficiency except in restricted cases, and their study focuses on possibility and impossibility results for elicitation with this restricted functionality.

I outline the framework here in brief; a detailed description, together with theoretical and empirical analysis, is provided in Chapter 5. There are three components: preference elicitation, winner determination, and pricing.

Preference elicitation. Each agent is assigned a proxy, which maintains a model of its agent's preferences. The model only partially agrees with the agent's preferences throughout the process, and is refined as needed.

Winner determination. The proxies' models are encoded in a language that allows for straightforward translation into a winner-determination MIP. This MIP can then be solved to obtain a tentative efficient allocation.

Pricing. Given the preference models together with an efficient allocation, the pricing engine computes market-clearing prices.

Each of these components is modular. The choice of techniques to use to quickly solve the MIP can be varied according to context. Similarly, the choice of algorithm used to compute market-clearing prices can be varied, and I propose two approaches in this dissertation (in Chapters 5 and 6). Some algorithms can perform winner determination and pricing in tandem. Finally, the learning algorithm a proxy uses to

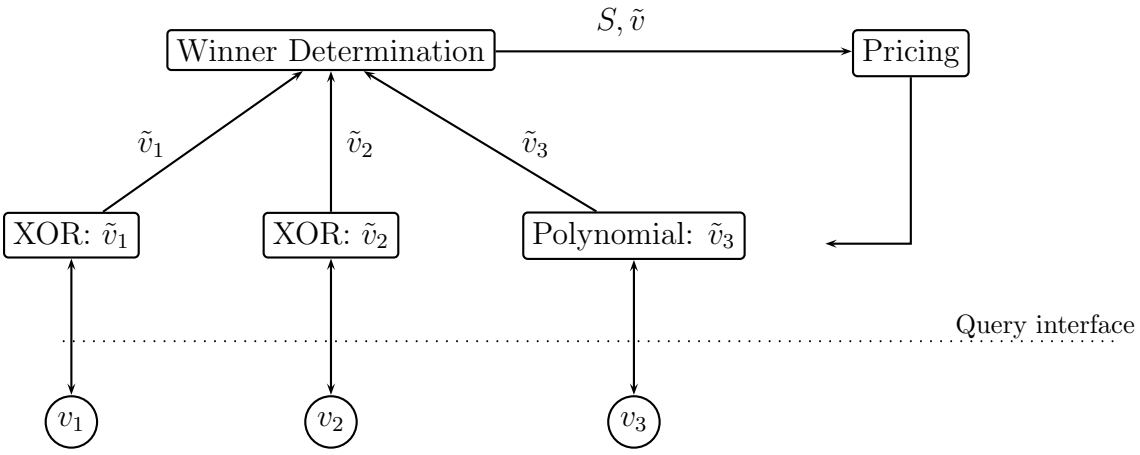


Figure 1.1: Sketch of the framework.

derive a model of its agent's preferences can also be adapted to the specific domain at hand, as long as it interacts with agents through an interface of value and demand queries.

A sketch of the framework is given in Figure 1.1. The proxies develop partial models $\tilde{v}_i$ of the agents' preferences v_i via queries. This can be done in parallel, asynchronously. At certain well-defined points, the elicitation is halted and an allocation S together with prices p are computed through the winner-determination and pricing engines. The allocation and prices are then returned to the agents via the proxies. If the prices do not clear the market for the agents, the elicitation restarts. This repeats until an efficient allocation together with market-clearing prices are reached. For the specific instantiations of the framework that I give in this dissertation, convergence is guaranteed.

This approach reduces the multi-agent elicitation problem to a single-agent learning problem. Also, the design is modular enough to accomodate several languages at

once, as Figure 1.1 suggests: if one agent's preferences are best modeled with XOR whereas another's are better modeled with Polynomials, the proxies can use these languages to recreate their respective agents' (partial) value information during the resource allocation process. No assumptions need to be made on the data structures the agents themselves use to encode their preferences, as long as the agents can respond to the required queries.

1.4 Incentives

As well as ensuring that preferences can be quickly elicited via queries, a multi-agent preference elicitation scheme must also ensure that agents will honestly respond to these queries. If the individual agents' interests are not aligned with the system-wide objective, they may try to game the process.

The efficient iterative auctions mentioned previously, as well as instantiations of the preference elicitation framework given in this dissertation, all proceed by presenting prices and each round, finally converging to market-clearing prices. In technical terms, market-clearing prices are called *competitive equilibrium* (CE) prices. In a competitive equilibrium, each agent's allocated bundle is most preferred at the quoted prices, and the allocation maximizes the auctioneer's revenue at the quoted prices. In this sense, demand equals supply and the market clears.

Since they are used as a halting criterion, it is important to understand under what conditions CE prices even exist. In our model, prices and preferences are symmetric: prices may also be defined over bundles, i.e. nonlinear, and prices may be differrent across agents, i.e. non-anonymous. In this case, CE prices always exist [13].

Since communicating nonlinear, non-anonymous prices is expensive, succinct "pricing languages" are needed in analogy to bidding languages. Alternatively, CE prices of lower-dimensionality may exist for certain restricted classes of agent preferences. Adapting the price structure when preference restrictions are met can lead to substantial communications savings during the auction. The question of the existence of CE prices in various environments is addressed in Chapter 3.

Protocols based on demand queries correctly compute an efficient outcome when agents are *myopic*, i.e. act as price-takers in each round, but in general we cannot expect this to be the case. The theory of mechanism design characterizes the kinds of payments that must be implemented to ensure agents report their preferences truthfully. A whole family of payment schemes exists with this property—the family of Groves mechanisms [42]—but within this family only the VCG mechanism [25, 42, 106] ensures that losing bidders pay nothing, and that the auctioneer maximizes his revenue under this constraint.

If the auction implements VCG payments, incentives are aligned with the system-wide objective. In a single-shot auction, this means truthful revelation is an optimal strategy no matter what the other agents do. In an iterative auction, this means myopic best-response bidding at each round is an optimal strategy if the other agents also follow this strategy. Chapter 2 provides a formal description of the VCG mechanism together with its incentive properties.

The *i*Bundle auction can be extended to compute VCG payments together with an efficient allocation. The idea, introduced by Parkes and Ungar [86], is to run the protocol further so that it converges to a *universal competitive equilibrium* (UCE)

instead of just a CE. In a UCE, the prices not only clear the market, but would also clear the market for any economy obtained by removing one agent from the system. Given UCE prices, it is possible to compute discounts to the final prices, so that the discounted price for each agent's allocated bundle corresponds to the agent's VCG payment. Mishra and Parkes [71] formalize the idea further and show that it can also be applied to the primal-dual auction of de Vries et al. [31]. In both extensions, the bidding language remains XOR, and the prices remain non-anonymous and are also represented using XOR.

The same idea applies to the framework proposed here. Once a CE has been reached, the elicitation can proceed with *univeral demand queries* rather than simple demand queries, which leads to a UCE. This does not complicate the query interface because universal demand queries can be reduced to a series of simple demand queries. The bidding language used by each proxy remains the same.

1.5 Model

There is a set of agents N and each agent is interested in acquiring items from a set M, which is held by a single seller. Let $n = |N|$ and $m = |M|$ be the numbers of agents and items. The items are indivisible, and the seller has no value for any item or bundle of items. Each agent i is described by a valuation function over bundles $v_i : 2^M \rightarrow \mathbf{R}_+$.

An *allocation* is a vector of bundles $R = (R_i)_{i \in N}$, where R_j is the bundle allocated to agent j. An allocation is *feasible* if $R_i \cap R_j = \emptyset$ for $i \neq j$. We denote the set of feasible allocations among agents N by $\Gamma(N)$, and similarly for subsets of N. A

partition is a set of bundles rather than a list of bundles, so that the assignment to the agents is not defined. We denote the set of feasible partitions of the items into $|N|$ bundles by $\Omega(N)$, and similarly for subsets of N.

If the allocation is R and agents are charged payments $q = (q_i)_{i \in N}$, then the utility to agent j is

$$u_j(R, q; v) = v_j(R_j) - q_j.$$

As denoted, utility functions are parametrized by the valuation profile $v = (v_i)_{i \in N}$, but this parameter will often be suppressed when clear from context.

There are several things to note about the form of the utility function. First, each agent knows its own valuation v_i, and its values do not change if it learns of any other agent's values, so this is a model with *private values*. Second, there are *no externalities*, meaning that an agent only cares about the bundle it acquires, and not what other agents obtain. Third, agents' utilities are *quasi-linear*, meaning that they can be denoted in a common currency, and utility can be transferred between agents in the form of payments.

Throughout, we also assume that valuations are: (1) *monotone*, i.e. for bundles S, T such that $S \subseteq T$, $v_i(S) \leq v_i(T)$; (2) *normalized*, i.e. the value for the empty set is $v_i(\emptyset) = 0$. We call valuations that satisfy these assumptions *general* valuations. Besides general valuations, the following subclasses will also be of interest. In our model, prices are also functions from bundles to non-negative real values, so these definitions apply equally to prices.

Additive An additive (also called 'linear') valuation satisfies

$$v(S) = \sum_{j \in S} v(j).$$

When agents have additive valuations, the allocation problem reduces to the problem of allocating each item efficiently in isolation, without regard to the other items. Hence the allocation problem here reduces to the single-item case.

Unit-demand A unit-demand valuation satisfies

$$v(S) = \max_{j \in S} v(j).$$

Unit demand is perhaps the simplest interesting family of valuations in a multi-item setting. The allocation problem in this instance is known as the *assignment problem*.

Single-minded A valuation is single-minded if there is a bundle S such that

$$v(T) = \begin{cases} v(S) & \text{if } T \supseteq S \\ 0 & \text{otherwise} \end{cases}$$

A bidder with a single-minded valuation is interested in acquiring all the items in S, and no more.

Superadditive A valuation is superadditive if for any two bundles S, T such that $S \cap T = \emptyset$,

$$v(S) + v(T) \leq v(S \cup T).$$

A valuation v is *subadditive* if the reverse inequality holds. Superadditivity captures the intuitive notion of complementarity: items are worth more together than separately. Similarly, subadditivity captures the notion of substitutability.

Supermodular A valuation is supermodular if for any two bundles S, T we have

$$v(S) + v(T) \leq v(S \cup T) + v(S \cap T).$$

A valuation v is *submodular* if the reverse inequality holds. Supermodular and submodular set functions are fundamental in combinatorial optimization. There are several alternate characterizations of these properties; in particular, submodularity (supermodularity) captures the notion of decreasing (increasing) marginal values with indivisible items.

Substitutes A valuation satisfies the *substitutes* condition if the following holds. If $S \in \arg\max v - p$ for a linear price function p, and $p' \geq p$ where p' is also linear, then there exists $T \in \arg\max v - p'$ such that $T \supseteq \{j \in S \mid p(j) = p'(j)\}$. In words, if the prices on some items increase, demand for the other items does not decrease. The substitutes condition was introduced by Kelso and Crawford [50], motivated by the problem of efficient allocation with indivisibilities. Substitutes valuations are submodular [72].

In traditional market models, prices are usually linear. The idea of nonlinear prices is a departure that was introduced by Bikhchandani and Ostroy [13]. It is important to distinguish between prices and payments. The seller defines prices over bundles to influence the agents' decisions; payments are actual transfers from the agents to the seller, so just a single payment is specified for each agent. If the seller quotes prices and an agent then selects a certain bundle, the agent's payment becomes the price of that bundle. In Chapter 2 we focus on schemes that use payments to affect the agents' incentives. Chapter 3 motivates and studies the use of prices.

The problem at hand is to find a feasible allocation R together with payments q that maximizes the total utility to the agents and seller:

$$\begin{aligned}\sum_{i\in N} u_i(R,q) + \sum_{i\in N} q_i &= \sum_{i\in N} [v_i(R_i) - q_i] + \sum_{i\in N} q_i \\ &= \sum_{i\in N} v_i(R_i).\end{aligned}$$

We see that payments cancel out, and the problem of finding an allocation and payments that maximize total utility is equivalent to the problem of finding an allocation that maximizes the total value to the agents. As mentioned earlier, this is commonly called the *combinatorial allocation problem* (CAP).

1.6 Outline

The following outlines the contents and contributions of each chapter. Chapter 2 provides background material on important concepts from microeconomics and mechanism design. It introduces the relevant game-theoretic solution concepts of dominant-strategies and Nash equillibrium, and describes the Groves family of mechanisms as well as the VCG mechanism. It also introduces *the core*, a central solution concept in cooperative game theory, with close ties to the notion of competitive equilibrium. Ideas from the theory of the core are used within the elicitation framework to derive CE prices.

Chapter 3 formally defines the notion of *competitive equilibrium* and reviews its fundamental properties. The chapter gives existence results for various forms of CE prices for different valuation classes. The notion of competitive equilibrium is also

related to the core and the VCG mechanism. The chapter is comprised of background material, except for the results relating to anonymous, nonlinear CE prices, which are novel contributions. The chapter concludes with an analysis of the communication requirements of efficient auctions, and of efficient auctions that compute VCG payments.

Chapter 4 examines representations for bids and prices in the presence of combinatorial preferences. It establishes the properties required of bidding and pricing languages, and introduces several languages. The main contributions of the chapter are learning algorithms for XOR and OR, and the novel language of pseudo-additive representations.

Chapter 5 describes the preference elicitation framework in detail. The chapter gives a proof of convergence for instantiations of the framework, and derives the complexity guarantees the framework provides in terms of the guarantees given by the underlying learning algorithms. There is also a detailed empirical evaluation.

Chapter 6 introduces a new iterative auction design that makes use of the pseudo-additive language. The algorithm has two variants: in one variant it is suitable as a pricing mechanism within the general elicitation framework, and in another it functions as a stand-alone ascending-bid auction.

Chapter 7 applies the principles of combinatorial auction design to the problem of Internet advertising. The first part of the chapter provides a survey of of the incentive, efficiency, and revenue properties of current advertising auctions for sponsored search. The second part proposes an expressive auction design for advertising over various Internet properties, including a new bidding language, together with

winner-determination and pricing algorithms, and a bidder-feedback mechanism.

Chapter 8 concludes with a discussion of the preference elicitation framework as well as other contributions in the dissertation, and outlines future research directions.

Chapter 2

Incentives

This chapter addresses the incentive issues that arise in the context of the combinatorial allocation problem when we asssume that agents are rational. To determine an efficient allocation, the seller must gather information on the agents' values, and base its decisions on this information. The agents, being rational, will choose to report their information in a way that maximizes their resulting utility. As a result, the allocation and payment rules must be carefully chosen to ensure that the right decision is made even if agents misreport some or all of their values.

The field of *mechanism design* studies precisely the problem of adjusting agent incentives so that the desired outcome occurs despite rational, manipulative behavior. In the context of the combinatorial allocation problem, fundamental methods in mechanism design such as the VCG mechanism reveal how to set payments to ensure agents will be truthful. Mas-Colell et al. [67] provide a good introduction to mechanism design and its central results.

Even if individual agents are motivated to truthfully reveal their values, the cho-

sen allocation and payments may still be problematic if certain coalitions of agents find that they could have been better off setting a different allocation and different payments among themselves. This motivates the concept of *the core*. The core captures the kind of divisions of gains from trade that ensure no coalition would have grounds to object to the outcome. In essence, then, the core captures the outcomes that can arise as the result of negotiation procedures among the buyers and seller.

As Ausubel and Milgrom [7] argue, core outcomes may be preferable to the Vickrey outcome in practice, because they guarantee a higher revenue for the auctioneer. Also, when goods are not substitutes, the VCG mechanism may be vulnerable to collusive behavior or shill bidding [92]. It is important then to understand when the Vickrey outcome will lie in the core.

Section 2.1 casts the combinatorial allocation problem as a mechanism design problem, and proposes the VCG mechanism as a way to ensure an efficient allocation is chosen. Along the way, relevant concepts from game theory and mechanism design are also introduced. Section 2.2 gives a formal treatment of the core, including a characterization specifically suited to the single-seller model, which has the advantage of making many results relating the core and the VCG mechanism quite transparent.

2.1 Dominant Strategies

The CAP can be cast in the framework of mechanism design. An *outcome* consists of a feasible allocation together with a specification of the payment required of each agent; formally, the set of outcomes is $\mathcal{O} = \Gamma(N) \times \mathbf{R}^N$. Note that an agent's valuation

summarizes all of its relevant information.[1] Let $\mathcal{V}$ denote the set of valuations from which agent values are drawn (e.g. general, supermodular, additive). The triplet $(N, \mathcal{O}, \mathcal{V}^N)$ describes the *environment*.

Recall that the seller values all bundles of items at zero, so its utility for an outcome (R, q) is simply the revenue it obtains, namely $\sum_{i \in N} q_i$. The seller's set of possible valuations is always this singleton, so the seller cannot misrepresent its private information. As a result, we can effectively ignore the seller when considering possible outcomes due to strategic behavior, as in the formulation of a "game" below.

A *mechanism* in our context consists of a set of actions $\mathcal{A}$ available to each player, an allocation rule $\xi^1 : \mathcal{A} \to \Gamma(N)$, and a payment rule $\xi^2 : \mathcal{A} \to \mathbf{R}^N$. The allocation and payment rules together form the outcome rule $\xi = (\xi^1, \xi^2)$ that maps action profiles into outcomes. The mechanism, together with the environment, induces a *game* $(N, \mathcal{A}^N, \mathcal{V}^N, \bar{u})$, consisting of a set of players, a set of action profiles, a set of valuation profiles, and a payoff function $\bar{u}$ mapping action profiles to payoffs. Given a realization of agent valuations $v = (v_i)_{i \in N}$, we have, for $a \in \mathcal{A}^N$,

$$\bar{u}_i(a; v) = u_i(\xi(a); v) = v_i(\xi_i^1(a)) - \xi_i^2(a),$$

according to our assumptions on the agents' utility functions. Because we have private values throughout, we will often suppress the parameter v from the utility function.

2.1.1 Solution Concept

A *strategy* is a map from valuations to actions, $s : \mathcal{V} \to \mathcal{A}$. A *solution concept* for a game specifies a set of strategy profiles that could emerge as the result of strategic

[1] In mechanism design parlance, an agent's valuations is its *type*, but we will not have any need for this more abstract terminology.

interaction among the agents. This in turn specifies a set of action profiles that can occur once valuations are realized, and hence gives a prediction of the set of possible outcomes. The central solution concept in our context is *dominant strategy equilibrium*.

A strategy profile s is a dominant-strategy equilibrium if, for any $i \in N$, $v \in \mathcal{V}^N$, and strategy profile s',

$$\bar{u}_i(s_i(v_i), s'_{-i}(v_{-i})) \geq \bar{u}_i(s'_i(v_i), s'_{-i}(v_{-i})).$$

In words, a strategy profile is a dominant-strategy equilibrium if, once valuations are realized, each agent's chosen action is a best-response no matter what actions the others choose.

In later chapters we will also make reference to *ex-post Nash equilibrium*, so we introduce it here in passing. A strategy profile s is an ex-post Nash equilibrium if, for any $i \in N$, $v \in \mathcal{V}^N$, and any strategy s'_i,

$$\bar{u}_i(s_i(v_i), s_{-i}(v_{-i})) \geq \bar{u}_i(s'_i(v_i), s_{-i}(v_{-i})).$$

In words, a strategy profile is an ex-post Nash equilibrium if, once valuations are realized, each agent's chosen action is a best-response to the others' actions.

A *social choice rule* (or simply 'choice rule') is a correspondence $F : \mathcal{V} \twoheadrightarrow \mathcal{O}$ that assigns a set of outcomes to each valuation profile. A mechanism *implements* a choice rule F if, for all possible realizations of valuations v, an equilibrium s of the induced game leads to an outcome $\xi(s(v)) \subseteq F(v)$. For instance, we might consider mechanisms that implement efficient outcomes with respect to realized valuations. The solution concept must be specified: if it is dominant-strategy equilibrium, we

say the mechanism implements the choice rule *in dominant strategies*. Similarly, if it is ex-post Nash equilibrium, the mechanism implements the choice rule *in ex-post Nash equilibrium*. Again, in this chapter, we focus on implementation in dominant strategies.

A mechanism is *direct* if $\mathcal{A} = \mathcal{V}$, i.e. if the agents must report a valuation. Otherwise, the mechanism is *indirect*. A direct mechanism is *incentive-compatible* if the strategy profile defined by $s_i(v_i) = v_i$ for all $i \in N$ is an equilibrium according to the relevant solution concept. In the special case where the solution concept is dominant-strategy equilibrium, we call the incentive-compatible mechanism *truthful*. The revelation principle states that for any mechanism, there is an equivalent incentive-compatible direct mechanism that implements the same outcomes. This principle is very useful as a theoretical tool for proving incentive properties of mechanisms.[2]

2.1.2 Implementation

Within this framework of mechanism design, we can now ask: what outcome rules can be implemented in dominant strategies? In light of the revelation principle, it is enough to ask what outcome rules can be implemented by a truthful direct mechanism. In the CAP, the allocation rule is essentially given (we seek an efficient allocation), while the payment rule is left to the discretion of the mechanism designer. The question is then how to select the payment rule to induce the agents to reveal

[2]In practice, however, it may still be preferable to implement the indirect mechanism rather than its incentive-compatible direct counterpart, because the computation and communication requirements of the mechanisms may be very different. If the valuation space is high-dimensional, for instance, then communicating a valuation is very costly. See Conitzer and Sandholm [26] for a discussion of these matters.

their values truthfully.

This question was answered by the seminal work of Vickrey [106], Clarke [25], and Groves [42] (VCG). Suppose the agents report valuation profile $\tilde{v}$. A *Groves mechanism* implements an efficient allocation R with respect to $\tilde{v}$, and charges payment

$$-\sum_{j \in N-i} \tilde{v}_j(R_j) + h_i(\tilde{v}_{-i}) \tag{2.1}$$

to agent i, where h_i is some arbitrary function of the reported valuations of agents other than i. The corresponding payoff to the agent is then

$$\left(v_i(R_i) + \sum_{j \in N-i} \tilde{v}_j(R_j) \right) - h_i(\tilde{v}_{-i}). \tag{2.2}$$

Note that the agent has no influence over the last term, while in the first it can only influence the choice of allocation R. The mechanism chooses an allocation that is efficient with respect to reported values, so the first term aligns the agent's incentives with the mechanism designer's objective of implementing an efficent outcome. It then follows that it is indeed a dominant strategy for agents to report values truthfully with such payoffs (see any of [54, 67, 69] for a rigorous argument). As a result, we will have $\tilde{v} = v$.

In fact, Green and Laffont [41] and Holmstrom [46] have shown that only payment rules of the form (2.1) successfully lead to dominant-strategy implementation of efficient outcomes, under the mild condition that the given valuation class is smoothly path connected.[3]

[3]A valuation class is smoothly path-connected if there is a continuous and differentiable path between any two valuations. It is simple to see that the classes of general, supermodular/submodular, superadditive/subadditive, and additive valuations are smoothly path-connected, because they are convex. The class of unit-demand valuations is also smoothly path-connected, because we can move from any valuation to the zero valuation, and then to any other. It is unclear whether this condition holds for the class of substitutes valuations, however.

The family of Groves mechanisms gives the designer some flexibility in choosing the functions h_i. One simple choice is $h_i(\tilde{v}_{-i}) = 0$ for all $i \in N$ and $\tilde{v}_{-i}$. Note, however, that this is very costly—the designer must payout the total value many times over. Let R' be an efficient allocation among agents $N - i$. The *VCG mechanism* is the Groves mechanism with

$$h_i(\tilde{v}_{-i}) = \sum_{j \in N-i} \tilde{v}_j(R'_j).$$

As a Groves mechanism, the VCG mechanism is truthful. The corresponding payment from agent i, called its *Vickrey payment*, is

$$\begin{aligned} \hat{q}_i &= \sum_{j \in N-i} v_j(R'_j) - \sum_{j \in N-i} v_j(R_j) && (2.3) \\ &= v_i(R_i) - \left(\sum_{j \in N} v_j(R_j) - \sum_{j \in N-i} v_j(R'_j) \right), && (2.4) \end{aligned}$$

which gives the agent a payoff of

$$\hat{\pi}_i = \sum_{j \in N} v_j(R_j) - \sum_{j \in N-i} v_j(R'_j), \qquad (2.5)$$

called the *Vickrey payoff*. Note that the Vickrey payoff is in fact independent of the choice of efficient allocations R and R'. The Vickrey payment, on the other hand, depends on the specific efficient allocation R selected because of the leading term in (2.4) (so the payment may vary depending on how ties are broken to select among efficient allocations).

Clearly, the Vickrey payoff (2.5) is non-negative no matter what valuation the agent reports, so the VCG mechanism is also *individually-rational*. This is a desirable property, because it ensures agents would never be unwilling to participate in the mechanism (note that an agent gets a payoff of zero if it does not participate, because

it is left with the empty set). In our context, the VCG mechanism is also (weak) budget-balanced, meaning that the total transfer to the seller is always non-negative. To see this, note that we necessarily have

$$\sum_{j \in N-i} v_j(R'_j) \geq \sum_{j \in N-i} v_j(R_j)$$

by the choice of R', so each agent's Vickrey payment 2.3 is non-negative.

There may be other truthful, individual-rational, and (weak) budget balanced mechanisms that implement efficient outcomes—these would necessarily be Groves mechanisms—but Krishna and Perry [55] show that the VCG mechanism in fact maximizes expected revenue among mechanisms with these properties. (The proof is technical and omitted.)

Theorem 1 [55] *Among all mechanisms for allocating items M to agents N that are efficient, incentive-compatible, and individually-rational, the VCG mechanism maximizes the expected payment of each agent.*

In light of this theorem, the VCG mechanism will be of particular interest among the class of Groves mechanisms in the rest of this thesis.

2.2 The Core

The CAP can also be viewed as a cooperative game with players $N \cup \{0\}$, where 0 denotes the seller. Define the *coalitional value function* w over subsets $L \subseteq N$ as

$$w(L) = \max_{R \in \Gamma(L)} \sum_{i \in L} v_i(R_i).$$

Here we have implicitly included the seller in the "coalition" along with the agents in L. In our model, coalitions that do not contain the seller have value zero. The coalitional value function is monotone and superadditive, because the agents' valuations are normalized, so (N, w) correctly defines a cooperative game (see e.g. Osborne and Rubinstein [77] for the basics of cooperative game theory).

A solution concept in this case specifies a set of possible payoffs $(\pi_0, \pi) \in \mathbf{R} \times \mathbf{R}^N$ to the players. A payoff vector can be construed as a division of the gains from trade among the agents and seller. The solution concept indirectly specifies a set of outcomes, namely those that give the agents payoffs π that agree with the solution concept.

A central solution concept in cooperative game theory is *the core*. In our context, a payoff vector $(\pi_0, \pi) \in \mathbf{R} \times \mathbf{R}^N$ is in the core, denoted $C(N)$, if

$$\pi_0 + \sum_{i \in N} \pi_i = w(N) \tag{2.6}$$

$$\pi_0 + \sum_{i \in L} \pi_i \geq w(L) \qquad \forall\, L \subseteq N \tag{2.7}$$

$$\sum_{i \in L} \pi_i \geq 0 \qquad \forall\, L \subseteq N \tag{2.8}$$

When restricting attention to a subset of agents L together with the seller, we similarly denote the core with respect to this new economy by $C(L)$. The core captures the notion of outcomes that are in a sense "stable". If an outcome leads to payoffs that are not in the core, then some coalition of agents, together with the seller, would have an incentive to reject it and instead negotiate a better outcome for itself. Hence if the agents and seller negotiate among themselves and decide on an outcome, we would expect it to yield core payoffs. A coalition consisting solely of buyers can of course

create a value of zero, so this is a lower bound for the total payoff of such coalitions, hence inequalities (2.8).

2.2.1 Alternate Characterization

Since we always have $\pi_0 = w(N) - \sum_{i \in N} \pi_i$, we can define the core in terms of constraints on the bidder payoffs only. Define the polytope $\tilde{C}(N)$ by

$$\sum_{i \in L} \pi_i \leq w(N) - w(N \backslash L) \qquad \forall\, L \subseteq N \tag{2.9}$$

$$\pi_i \geq 0 \qquad \forall\, i \in N \tag{2.10}$$

As an alternate characterization of the core, we have the following.

Lemma 1 *$(\pi_0, \pi) \in C(N)$ if and only if $\pi \in \tilde{C}(N)$ and $\pi_0 = w(N) - \sum_{i \in N} \pi_i$.*

Proof. Let $(\pi_0, \pi) \in C(N)$. By (2.6), $\pi_0 = w(N) - \sum_{i \in N} \pi_i$. Also,

$$\begin{aligned} \sum_{i \in L} \pi_i &= \left(\pi_0 + \sum_{i \in N} \pi_i \right) - \left(\pi_0 + \sum_{i \in N} \pi_i - \sum_{i \in L} \pi_i \right) \\ &= w(N) - \left(\pi_0 + \sum_{i \in N \backslash L} \pi_i \right) \\ &\leq w(N) - w(N \backslash L), \end{aligned}$$

where the second equality follows from (2.6) and the inequality from (2.7). Hence $\pi \in \tilde{C}(N)$.

Now assume $\pi \in \tilde{C}(N)$ and $\pi_0 = w(N) - \sum_{i \in N} \pi_i$. By the latter (2.6) holds. Also, for $L \subseteq N$,

$$\begin{aligned} \pi_0 + \sum_{i \in L} \pi_i &= \pi_0 + \sum_{i \in N} \pi_i - \sum_{i \in N \setminus L} \pi_i \\ &= w(N) - \sum_{i \in N \setminus L} \pi_i \\ &\geq w(N) - w(N) + w(L) \\ &= w(L), \end{aligned}$$

where the inequality follows from (2.9); this establishes (2.7). Because of (2.10), the total payoff to a coalition that does not contain the seller is always zero, so (2.8) holds. Hence $(\pi_0, \pi) \in C(N)$. □

Setting $\pi_i = 0$ for each agent $i \in N$ and $\pi_0 = w(N)$, we see from Lemma 1 that these payoffs are in the core, and hence that the core is non-empty in our model. This core payoff point corresponds to the case where the seller extract all the available value.

2.2.2 Relation to Vickrey Payoffs

We saw in the previous section that a mechanism is incentive-compatible in dominant strategies if it implements an outcome that gives agents their Vickrey payoffs. It is then natural to ask: when are Vickrey payoffs in the core? When we implement Vickrey payoffs and they are also in the core, agents are motivated to report their valuations truthfuly, and no coalition has any incentive to reject the outcome. We first note a general relationship between Vickrey and core payoffs.

Proposition 1 [80, 69]

$$\hat{\pi}_i = w(N) - w(N-i) = \max_{\pi \in \tilde{C}(N)} \pi_i$$

Proof. The first equality follows from the definition of Vickrey payoffs (2.5). Let $\pi \in \tilde{C}(N)$. Inequality (2.7) with $S = \{i\}$ shows that $\pi_i \leq w(N) - w(N-i)$. Taking $\pi_i = \hat{\pi}_i$ and $\pi_j = 0$ for all other bidders, we see that π satisfies inequalities (2.9) because w is monotone, and also (2.10) because Vickrey payoffs are non-negative. Hence we in fact have $\hat{\pi}_i = \max_{\pi \in \tilde{C}(N)} \pi_i$. □

It turns out that submodularity of the coalitional value function w is key to the relationship between Vickrey and core payoffs. Here is a useful alternate characterization of submodularity. (The characterization is standard so the proof is omitted.)

Lemma 2 [53, 107] *Assume w is monotone. Then w is submodular if and only if for all $K, L \subseteq M$,*

$$w(L) - w(K) \quad \geq \quad \sum_{i \in L \setminus K} [w(L) - w(L-i)] \tag{2.11}$$

Inequalities (2.11) with the restriction that $L = N$ form a condition called "buyers are substitutes" [13]. Under this condition, $\tilde{C}(N)$ has a particularly simply description:

$$\pi_i \leq w(N) - w(N-i) \qquad \forall\, i \in N \tag{2.12}$$

$$\pi_i \geq 0 \qquad \forall\, i \in N \tag{2.13}$$

To see this, note that by summing inequalities (2.12) over all $i \in K$ for any $K \subseteq N$ yields

$$\sum_{i \in K} \pi_i \leq \sum_{i \in K} [w(N) - w(N-i)] \leq w(N) - w(N \setminus K)$$

where the last inequality follows from the condition that buyers are substitutes. Hence inequalities (2.12) imply inequalities (2.9) under this condition. Under the characterization given by (2.12) and (2.13), it is easy to see that $\tilde{C}(N)$ is a lattice with respect to the usual meet and join operations on real vectors (i.e. taking component-wise minimums and maximums).

Proposition 2 [13] *If buyers are substitutes, then $\tilde{C}(N)$ is a lattice.*

Proof. Let $\pi, \pi' \in \tilde{C}(N)$. Since (2.12) and (2.13) hold for both π_i and π'_i, for all $i \in N$, they also hold for $\max\{\pi_i, \pi'_i\}$ and $\min\{\pi_i, \pi'_i\}$. □

Since the core is a lattice under "buyers are substitutes", it follows from Proposition 1 that this condition also implies the Vickrey payoff point is in the core: simply take the meet of the elements that give the agents their Vickrey payoffs. This reasoning does not work when buyers are not substitutes, which leads us to the central result relating the core and Vickrey payoffs.

Theorem 2 [13] *Vickrey payoffs are in the core if and only if buyers are substitutes.*

Proof. Since the Vickrey outcome is efficient, the Vickrey payoffs satisfy $\hat{\pi}_0 + \sum_{i \in N} \hat{\pi}_i = w(N)$. Thus, by Lemma 1, the condition that Vickrey payoffs are in the core and the condition that buyers are substitutes are both equivalent to

$$w(N) - w(N \backslash L) \geq \sum_{i \in L} [w(N) - w(N - i)]$$

for all $L \subseteq N$. □

Note that when buyers are substitutes, the Vickrey payoff point is the minimal element of the core viewed as a lattice—agents receive their maximum possible payoffs

at this point, and the seller receives its minimum possible revenue. The core payoff point mentioned earlier, where the seller extracts all the value, is the maximal element of the core viewed as a lattice.

The discussion so far has centered on properties of the coalitional value function w, but the primitives in our model are the agent valuations. Ausubel and Milgrom [7] identify a crucial property of valuations that leads to a submodular coalitional value function. (The proof is very technical and omitted.)

Theorem 3 [7] *Let $\mathcal{V}$ be a class of valuations that contains the additive valuations. Then the coalitional value function w corresponding to every profile of valuations drawn from $\mathcal{V}$ is submodular if and only if each valuation in $\mathcal{V}$ satisfies the substitutes condition.*

Of course, to ensure that Vickrey payoffs are in the core, we only need the "buyers are substitutes" condition, and the condition that w is submodular is stronger. Currently, no weaker property than the substitutes condition is known that would lead to the "buyers are substitutes" condition.

By Theorems 2 and 3, Vickrey payoffs are in the core for substitutes valuations. This is also the case for additive and unit-demand valuations, because these are subclasses of the substitutes valuations. Theorem 3 shows that this is not necessarily the case for profiles drawn from the other valuation classes mentioned in Section 1.5: general, submodular/supermodular, and subadditive/superadditive. These results are summarized in Table 2.1.

These results are quite negative, because we will often be interested in classes other than the substitutes valuations. In such situations, the designer must choose

	VPP $\in$ Core
Linear	✓
Unit-Demand	✓
Substitutes	✓
Supermodular	×
Superadditive	×
Submodular	×
Subadditive	×
General	×

Table 2.1: This table lists, for each class of valuations, whether the Vickrey payoff point (VPP) necessarily lies in the core. The results all follow from Theorem 3 due to Ausubel and Milgrom [7], except for the case of unit-demand valuations, which is due to Leonard [62], and the linear case, which is immediate.

whether it is more important to implement Vickrey payoffs, so that agents have no incentives to manipulate the system, or to implement core payoffs, so that the outcome is robust to deviations from coalitions. As one approach to managing this tradeoff, Parkes, Kalagnanam, and Eso [84] and Day and Milgrom [30] study the problem of implementing core payoffs that are in a sense "closest" to the Vickrey payoff point, the idea being that these core payoffs minimize the agents' incentives to misreport their preferences. In this thesis, we will look at ways to implement both the Vickrey outcome, and core outcomes that are close to the Vickrey outcome.

Chapter 3

Competitive Equilibrium

Iterative auctions, including the preference elicitation framework proposed in this dissertation, operate by quoting prices at each round to drive the bidding, eventually converging to market-clearing prices. Formally, market-clearing prices are referred to as *competitive equilibrium prices*. In the package assignment model, the auctioneer may choose to quote *nonlinear* prices, i.e. a distinct price for each bundle that is not simply the sum of item prices. The auctioneer may also introduce price discrimination so that different agents face different price vectors. Depending on the structure of prices chosen, competitive equilibrium prices may or may not exist when items are indivisible.

Because competitive equilibrium is used as a halting criterion, it is important to understand when competitive equilibrium prices with the relevant structure exist. The main contributions of this chapter relate to superadditive valuations; the remainder serves as background. When agents have superadditive valuations, the auctioneer does not need to introduce price discrimination to ensure existence of a competitive

equilibrium. Superadditive valuations correspond to the case where items are complements, a common regime in package auctions (see e.g. virtually any chapter in Cramton et al. [28]). Since package auctions find application in the public sector, "fair" pricing that does not discriminate among bidders is often a constraint; it is also often a concern in procurement settings [10].

Section 3.1 gives a formal definition of the concept of competitive equilibrium, and Section 3.2 surveys the properties of allocations and prices that arise in competitive equilibirium. Section 3.3 provides a detailed study of the structure of equilibrium prices for various classes of valuations. Section 3.4 relates the payoffs that arise in competitive equilibrium to the concept of the core, and Section 3.5 gives conditions under which Vickrey payments can arise in equilibrium. Section 3.6 relates the communication requirements of computing Vickrey payments to the extended notion of a *universal competitive equilibrium*.

3.1 Definition

We allow for nonlinear prices, so that a distinct price may be quoted for each bundle. Bikhchandani and Ostroy [13] also introduced the idea of *non-anonymous* prices, to allow for situations where different agents are quoted different prices for the same bundles. Hence, instead of quoting a single price function over bundles, we may quote a different price function p_i to each agent. If $p_i = p_j$ for any two agents i, j then we say the prices are *anonymous*.[1] We identify three orders or pricing:

[1]This is the sole source of asymmetry between valuations and prices in our model. It is not of much interest to talk of "anonymous" valuation profiles, because we usually expect valuations to differ among agents.

1. Linear and anonymous.

2. Nonlinear and anonymous.

3. Nonlinear and non-anonymous.

Very little is known about linear and non-anonymous prices, and we do not consider them in this thesis. In some cases, it is also natural to consider prices of order 0: linear prices where all items have the same price, so that only the size of a bundle is relevant when determining its price.

An allocation R together with prices $p = (p_i)_{i \in N}$ constitute a *competitive equilibrium* (CE) if for all $R' \in \Gamma(N)$,

$$v_i(R_i) - p_i(R_i) \geq v_i(R'_i) - p_i(R'_i) \qquad \forall\, i \in N \tag{3.1}$$

$$\sum_{i \in N} p_i(R_i) \geq \sum_{i \in N} p_i(R'_i) \tag{3.2}$$

In a competitive equilibrium, each agent's allocated bundle maximizes the agent's utility at the given prices, and the chosen allocation also maximizes the seller's revenue at the given prices. In this sense, supply equals demand and the market clears. We call R a *competitive equilibrium allocation* and p *competitive equilibrium prices*. We say that the CE prices *support* the CE allocation.

3.2 Properties

Let $\mathcal{E} \subseteq \Gamma(N)$ denote the set of CE allocations and $\mathcal{P} \subseteq \mathbf{R}^{N \times 2^N}$ the set of CE prices (i.e. allocations and prices that appear in some competitive equilibrium). The set of competitive equilibria is in fact a product set.

Proposition 3 [67] *The set of competitive equilibria is $\mathcal{E} \times \mathcal{P}$.*

Proof. Let (R, p) be a competitive equilibrium, and let (R', p') be another. We have

$$\begin{aligned} \sum_{i \in N} v_i(R_i) &= \sum_{i \in N} [v_i(R_i) - p'_i(R_i)] + \sum_{i \in N} p'_i(R_i) \\ &\leq \sum_{i \in N} [v_i(R'_i) - p'_i(R'_i)] + \sum_{i \in N} p'_i(R'_i) \qquad (3.3) \\ &= \sum_{i \in N} v_i(R'_i). \end{aligned}$$

By an identical argument, $\sum_{i \in N} v_i(R'_i) \leq \sum_{i \in N} v_i(R_i)$, and so $\sum_{i \in N} v_i(R_i) = \sum_{i \in N} v_i(R'_i)$. Hence inequality (3.3) holds with equality, and each R_i is maximizes i's utility at prices p'_i, while R maximizes revenue at prices p. This shows that (R, p') is a competitive equilibrium. By an identical argument, (R', p) is a competitive equilibrium. Therefore the set of competitive equilibria is the product set $\mathcal{E} \times \mathcal{P}$. □

Given this lemma, it makes sense to speak of "competitive equilibrium prices" without reference to any particular allocation; this is in contrast to Vickrey payments. Like the allocations that result from the VCG mechanism or from core outcomes, CE allocations are efficient. This is the "first fundamental theorem of welfare economics," specialized to our single-seller model with indivisibilities.

Theorem 4 [13, 67] *Competitive equilibrium allocations are efficient.*

Proof. Let R be a CE allocation, and let p be corresponding CE prices. Given a feasible allocation R', summing inequalities (3.1) and (3.2) yields $\sum_{i \in N} v_i(R_i) \geq \sum_{i \in N} v_i(R'_i)$. Since R' was arbitrary, R is efficient. □

In fact, if an allocation is efficient, then there are CE prices that support it, as the following theorem shows. This is the "second fundamental theorem of welfare

economics," adapted to our model. It implies, in particular, that $\mathcal{E}$ is exactly the set of efficient allocations.

Theorem 5 [13] *For every efficient allocation R, there exist prices p such that (R, p) is a competitive equilbrium.*

Proof. Define prices $p_i(S) = v_i(S)$ for all $S \subseteq 2^M$ and $i \in N$. An agent receives utility zero from any bundle, so R_i is trivially utility-maximizing to agent i, for all $i \in N$. Since the revenue of an allocation equals its value, R must then maximize revenue because it is efficient. □

The proof of Theorem 5 shows that CE prices of order 3 always exist. Such prices are clearly costly to quote, in terms of communication. It is therefore useful to ask whether competitive equilibria exist with lower order prices.

3.3 Existence

Whether CE prices of lower order exist will depend on the specific valuation class being considered. Kelso and Crawford [50] show that order 1 CE prices exist when agents have substitutes valuations. This is then also the case when agents have additive or unit-demand valuations.

When agents have superadditive valuations, order 2 prices exist. This result was first given by Parkes [79], who provides an algorithmic proof. We give here a simpler, constructive, and non-algorithmic proof.

Theorem 6 [79] *There exist anonymous, nonlinear competitive equilibrium prices if agents have superadditive valuations.*

Proof. Let R be an efficient allocation. Define $p(S) = \max_{i \in N} v_i(S)$ for $S \subseteq M$. We claim these are CE prices. First note, that $p(R_i) = v_i(R_i)$. To see this, assume to the contrary that $p(R_i) > v_i(R_i)$. Then there is some $j \neq i$ such that $v_j(R_i) > v_i(R_i)$. We then have

$$v_i(R_i) + v_j(R_j) < v_j(R_i) + v_j(R_j) \leq v_j(R_i \cup R_j) + v_i(\emptyset),$$

where the second inequality follows from the fact that v_j is superadditive and $R_i \cap R_j = \emptyset$ (since they are part of a feasible allocation), and the fact that v_i is normalized. Hence if in allocation R we replace R_i with $\emptyset$ and R_j with $R_i \cup R_j$, we obtain an allocation with strictly greater value than R, contradicting its efficiency.

We must then have $p(R_i) = v_i(R_i)$, and the payoff to each agent under these prices is 0. By our price construction, the utility from any other bundle to any agent is at most 0. Hence the prices ensure the allocated bundles are utility-maximizing for the agents.

Now among revenue-maximizing allocations, choose an allocation R' that maximizes the number of agents that receive nothing. We claim that we cannot have two nonempty bundles R'_i and R'_j such that $p(R'_i) = v_k(R'_i)$ and $p(R'_j) = v_k(R'_j)$; i.e. the prices of both bundles cannot be derived from the same agent's valuation. If this were the case, we would have

$$p(R'_i) + p(R'_j) = v_k(R'_i) + v_k(R'_j) \leq v_k(R'_i \cup R'_j) \leq p(R'_i \cup R'_j) + p(\emptyset)$$

by the superadditivity of v_k and the definition of p. We see then that replacing R'_i with $R'_i \cup R'_j$ and R'_j with $\emptyset$ would result in an allocation with weakly greater revenue, but with one more agent receiving nothing, which would contradict our original choice of R'.

If $R'_i \neq \emptyset$, reassign the bundle so that $p(R'_i) = v_i(R'_i)$. By our arguments above, this is a valid reassignment of such bundles. The remaining (empty) bundles can be reassigned to the remaining agents arbitrarily, and we then have $p(R'_i) = v_i(R'_i)$ for all $i \in N$. Because prices are anonymous, this reassignment does not change the revenue, and so we can assume without loss of generality that this was the original assignment. But note now that the revenue from R' is exactly its total value. The revenue from R is also its total value, as we saw above, so the revenue from R is at least the revenue from R', because R is efficient. Because R' maximizes revenue, so then does R.

The constructed prices ensure that each bundle in R maximizes its respective agent's utility, and that the allocation maximizes revenue to the seller. Therefore they are CE prices. □

It follows that order 2 prices exist for single-minded valuations as well. Note that when a valuation is normalized, it is superadditive if it is supermodular. Hence order 2 prices exist for supermodular valuations as well. In fact, when there are just two agents with supermodular valuations, then order 1 CE prices exist, but this fact does not extend to more agents [32].

The following result provides a converse of sorts to Theorem 6. There may be specific valuation profiles that allow for anonymous CE prices even though some valuations are not superadditive. However, this cannot be guaranteed for all valuation profiles unless valuations are only draw from the class of superadditive valuations.[2]

[2]Theorem 7 is analogous to Ausubel and Milgrom's result for substitutes valuations [7]. They show that the class of substitutes valuations is the largest class containing the additive valuations that guarantees that order 1 CE prices will exist for all valuation profiles. Their result applies when there are at least 4 bidders.

Theorem 7 *Suppose that $\mathcal{V}$ contains the class of single-minded valuations, and that $N \geq 3$. Then nonlinear, anonymous competitive equilibrium prices exist for every profile of valuations drawn from $\mathcal{V}^N$ only if every valuation in $\mathcal{V}$ is superadditive.*

Proof. Assume there is a valuation $v_1 \in \mathcal{V}$ that is not superadditive. Then there exist nonempty $S, S' \subseteq M$ such that $S \cap S' = \emptyset$ and $v_1(S) + v_1(S') > v_1(S \cup S')$. Let v_2 be a single-minded valuation with $v_2(T) = v_1(S \cup S')$ for $T \supseteq S \cup S'$ and $v_2(T) = 0$ otherwise. Finally, let v_3 be a single-minded valuation with $v_3(T) = v_1(M) + v_2(M)$ for $T \supseteq M \backslash (S \cup S')$ and $v_3(T) = 0$ otherwise. The valuations of any remaining agents are set to 0 over all bundles.

If agent 3 is not given a superset of $M \backslash (S \cup S')$, no value greater than $v_1(M) + v_2(M)$ can be achieved, so there exists an efficient allocation where agent 3 gets such a superset. On the other hand, giving the agent more than items $M \backslash (S \cup S')$ cannot add any value, so there is an efficient allocation where it receives exactly these items. To allocate the remaining items $S \cup S'$ efficiently, note that agent 2 only gets positive value if it obtains all these items, so it is efficient to either give the agent either all these items or none of them. We see then that allocation $(\emptyset, S \cup S', M \backslash (S \cup S'))$ is efficient.

Assume there exist anonymous prices p that support this allocation. Because $\emptyset$ maximizes agent 1's utility at these prices, we have

$$v_1(S) - p(S) \leq 0 \tag{3.4}$$

$$v_1(S') - p(S') \leq 0 \tag{3.5}$$

$$v_1(S \cup S') - p(S \cup S') \leq 0 \tag{3.6}$$

Agent 2 must prefer its bundle $S \cup S'$ to the empty set, so we also have

$$v_2(S \cup S') - p(S \cup S') \geq 0. \tag{3.7}$$

From (3.6) and (3.7) we see that $p(S \cup S') = v_1(S \cup S') = v_2(S \cup S')$. We then have

$$\begin{aligned} p(S) + p(S') &\geq v_1(S) + v_1(S') \\ &> v_1(S \cup S') \\ &= p(S \cup S'), \end{aligned}$$

where the first inequality follows from (3.4) and (3.5). But this means that the revenue from allocation $(S, S', M \backslash (S \cup S'))$ is strictly greater than the revenue from $(\emptyset, S \cup S', M \backslash (S \cup S'))$; so prices p cannot in fact support the latter, which gives us a contradiction. □

The following example shows that order 1 CE prices do not necessarily exist when agents have single-minded valuations, and hence this is also the case for supermodular and superadditive valuations.[3]

Example. The set of agents is $N = \{1, 2, 3, 4\}$ and the set of items is $M = \{a, b, c\}$. The agents' valuations are single-minded. Agent 1 wants *abc* and values it at 4. Agent 2 wants *ab* and values it at 3. Agent 3 wants *bc* and values it at 3. Agent 4 wants *ac* and values it at 3. It is efficient to give *abc* to agent 1 and nothing to the others. Assume there exist order 1 CE price that support this allocation. Since agents 2, 3,

[3]The example is for 3 items and 4 agents. If there are less than 3 items or less than 3 agents, it can be shown that order 1 CE prices do exist if all valuations are single-minded. So the example has the smallest possible parameter settings.

and 4 receive nothing, we must have

$$\begin{aligned} p(a) + p(b) &\geq 3 \\ p(b) + p(c) &\geq 3 \\ p(a) + p(c) &\geq 3 \end{aligned}$$

from which is follows that

$$p(a) + p(b) + p(c) \geq 9/2.$$

But in order for *abc* to maximize agent 1's utility, its price must be below 4. So we have reached a contradiction, and order 1 prices cannot exist.

The remaining question is whether there exist linear or anonymous CE prices when agents have subadditive or submodular valuations. The following example shows that only order 3 prices can guarantee the existence of a CE with these classes of valuations.[4] Naturally, this is then also the case with general valuations.

Example. The set of agents is $N = \{1, 2\}$ and the set of item is $M = \{a, b, c\}$. The agents' valuations are given in the following table.

	a	b	c	ab	ac	bc	abc
1	5	2	5	6	6	7	7
2	2	5	5	6	7	6	7

It is simple but tedious to check that these valuations are submodular. Here allocating a to agent 1 and bc to agent 2 is efficient. Assume there are order 2 CE prices p that

[4]The example shows that order 3 prices may be required to guarantee the existence of CE prices when there are 3 items and 2 agents with submodular valuations. If there are just 2 items, then it can be shown that order 2 CE prices exist. This case is quite trivial, however, so the proof is omitted.

	CE exists	References
Linear	1	Immediate
Unit-Demand	1	Koopmans and Beckmann [52]
Substitutes	1	Kelso and Crawford [50]
Single-minded	2	Parkes [79], this work
Supermodular	2	Parkes [79], this work
Superadditive	2	Parkes [79], this work
Submodular	3	Bikhchandani and Ostroy [13], this work
Subadditive	3	Bikhchandani and Ostroy [13], this work
General	3	Bikhchandani and Ostroy [13], this work

Table 3.1: This table lists, for each class of valuations, the lowest order of prices that ensures that competitive equilibrium prices exist. The references give first the work that shows a certain order of CE prices exists for the given valuation class, and then the work that gives an example to show a lower order is not sufficient (except of course for order 1 prices).

support this allocation. Since the agents' allocations maximize their utilities at these prices, we have

$$\begin{aligned} 5 - p(a) &\geq 7 - p(bc) \\ 6 - p(bc) &\geq 5 - p(b) \\ 6 - p(bc) &\geq 7 - p(ac) \end{aligned}$$

Summing these inequalities and rearranging, we find that

$$p(b) + p(ac) > p(a) + p(bc)$$

which contradicts the fact that allocation (a, bc) should maximize the seller's revenue at prices p. Hence order 2 CE prices do not exist. Order 1 prices are also anonymous, so these do not exist either. The conclusions on the existence of various orders of CE prices are summarized in Table 3.1.

3.4 Relation with the Core

We now turn to the relationship between competitive equilibrium and the core. Let $\pi_i = v_i(R_i) - p_i(R_i)$ for $i \in N$ and $\pi_0 = \sum_{i \in N} p_i(R_i)$ be the payoffs obtained in competitive equilibrium (R, p). We call these *competitive payoffs*. The following theorem can be construed as a strengthening of Theorem 4, because core payoffs can only arise from efficient outcomes.

Theorem 8 [67, 77] *Competitive payoffs are in the core.*

Proof. Let (R, p) be a competitive equilibrium. Let $\pi_i = v_i(R_i) - p_i(R_i)$ and $\pi_0 = \sum_{i \in N} p_i(R_i)$ be the corresponding competitive payoffs. For any coalition of agents L, let R' be a corresponding efficient allocation among them, and let $R'_i = \emptyset$ for $i \notin L$. Then,

$$w(L) = \sum_{i \in L} v_i(R'_i) = \sum_{i \in L} [v_i(R'_i) - p_i(R'_i)] + \sum_{i \in N} p_i(R'_i) \leq \sum_{i \in L} \pi_i + \pi_0, \tag{3.8}$$

where the inequality follows from (3.1) and (3.2). We have

$$\sum_{i \in N} \pi_i + \pi_0 = \sum_{i \in N} v_i(R_i) \leq w(N),$$

and combined with (3.8) for $L = N$, the inequality holds with equality. Hence (π_0, π) is in the core. □

Since competitive payoffs are in the core, might all core payoffs be competitive? If this is the case, we say that *the core can be priced*. As with the question of the existence of competitive equilibrium, whether the core can be priced depends on the order of prices used. Given core payoffs $(\pi_0, \pi) \in C(N)$, Bikhchandani and Ostroy [13]

show that prices

$$p_i(S) = \max\{v_i(S) - \pi_i, 0\}$$

yield a competitive equilibrium (R, p)—where R is any efficient allocation—that gives the agents payoffs π and the seller revenue π_0. Hence the core can always be priced using order 3 prices.

The proof of Theorem 6 shows that payoffs $\pi_i = 0$ for $i \in N$ and $\pi_0 = w(N)$, i.e. the seller-optimal core outcome, can be priced using order 2 prices when agents have superadditive valuations. This suggests that order 2 prices might actually price the core when agents have superadditive valuations, and this is indeed the case. First a useful lemma.

Lemma 3 *Let R be an efficient allocation, and let $\pi \in \tilde{C}(N)$. If all agents have superadditive valuations, $v_i(R_i) - \pi_i \geq v_j(R_i) - \pi_j$ for all $i, j \in N$.*

Proof. Consider the allocation where agent i obtains $\emptyset$, j obtains $R_j \cup R_i$, and every other agent k receives R_k as before. Note that

$$\begin{aligned} w(N-i) &\geq v_j(R_i \cup R_j) + \sum_{k \neq i,j} v_k(R_k) \\ &\geq v_j(R_i) + v_j(R_j) + \sum_{k \neq i,j} v_k(R_k) \end{aligned} \tag{3.9}$$

where the first inequality follows from the definition of w, and the second from the fact that v_j is superadditive. We then have

$$\begin{aligned} \pi_i &\leq w(N) - w(N-i) \\ &\leq v_i(R_i) - v_j(R_i) \end{aligned}$$

where the first inequality follows from Proposition 1, and the second from (3.9) above and the fact that $w(N) = \sum_{k \in N} v_k(R_k)$. Hence $v_i(R_i) - \pi_i \geq v_j(R_i)$, and since $\pi_j \geq 0$, it follows that $v_i(R_i) - \pi_i \geq v_j(R_i) - \pi_j$. □

We are now ready to prove our central theorem on order 2 prices. The line of argument is very similar to the proof of Theorem 6, which follows from the next theorem as a corollary.

Theorem 9 *The core can be priced with anonymous, nonlinear prices if all agents have superadditive valuations.*

Proof. Let R be an efficient allocation, and let $(\pi_0, \pi) \in C(N)$. Note that $w(N) = \sum_{i \in N} v_i(R_i)$. For all $S \subseteq 2^M$, let

$$p(S) = \max \left\{ \max_{i \in N} \{v_i(S) - \pi_i\}, 0 \right\}.$$

For each $i \in N$, we have

$$\begin{aligned} v_i(R_i) &= \sum_{j \in N} v_j(R_j) - \sum_{j \neq i} v_j(R_j) \\ &\geq w(N) - w(N - i) \\ &\geq \pi_i, \end{aligned}$$

where the first inequality follows from the definition of w, and the last from Proposition 1. Hence $v_i(R_i) - \pi_i \geq 0$ for all i. Because R is efficent and π is in the core, it then follows by Lemma 3 that $p(R_i) = v_i(R_i) - \pi_i$. Hence $v_i(R_i) - p(R_i) = \pi_i$, and for all $S \subseteq M$, $v_i(S) - p(S) \leq v_i(S) - [v_i(S) - \pi_i] = \pi_i$ by the definition of p. Therefore, buyers maximize their utility at these prices and indeed get payoffs π.

Note that $\sum_{i \in N} p(R_i) = \sum_{i \in N} v_i(R_i) - \sum_{i \in N} \pi_i = w(N) - \sum_{i \in N} \pi_i = \pi_0$. Among revenue-maximizing allocations, choose an allocation R' that maximizes the number

of agents that receive nothing. We claim that we cannot have two nonempty bundles R'_i and R'_j such that $p(R'_i) = v_k(R'_i) - \pi_k$ and $p(R'_j) = v_k(R'_j) - \pi_k$; i.e. the prices of both bundles cannot be derived from the same agent's valuation and core payoff. If this were the case, we would have

$$p(R'_i) + p(R'_j) = v_k(R'_i) + v_k(R'_j) - 2\pi_k \leq v_k(R'_i \cup R'_j) - \pi_k \leq p(R'_i \cup R'_j) + p(\emptyset)$$

by the superadditivity of v_k and the definition of p, and the fact that $\pi_k \geq 0$. We see then that replacing R'_i with $R'_i \cup R'_j$ and R'_j with $\emptyset$ would result in an allocation with weakly greater revenue, but with one more agent receiving nothing, which would contradict our original choice of R'.

If $R'_i \neq \emptyset$ and $p(R'_i) > 0$, reassign the bundle so that $p(R'_i) = v_i(R'_i) - \pi_i$. By our arguments above, this is a valid reassignment of such bundles. The remaining bundles with $p(R'_i) = 0$ can be reassigned to the remaining agents arbitrarily. Because prices are anonymous, this reassignment does not change the revenue, and so we can assume without loss of generality that this was the original assignment. Let N' be the agents that receive a bundle with positive price under allocation R'. The revenue from R' is then

$$\begin{aligned}
\sum_{i \in N} p(R'_i) &= \sum_{i \in N'} [v_i(R'_i) - \pi_i] \\
&\leq w(N') - \sum_{i \in N'} \pi_i \\
&\leq \pi_0 + \sum_{i \in N'} \pi_i - \sum_{i \in N'} \pi_i \\
&= \sum_{i \in N} p(R_i).
\end{aligned}$$

where the second inequality follows from the fact that (π_0, π) is in the core. Since R' was revenue-maximing at prices p, so is R.

The constructed prices ensure that each bundle in R maximizes its respective agent's utility, and that the allocation maximizes revenue to the seller. Therefore they are CE prices which lead exactly to core payoffs (π_0, π). □

In a single-item setting, it is straightforward to see that the core can be priced with order 1 prices, and this extends to additive valuations. The following example shows that order 2 prices do not necessarily price the core when agents have unit-demand valuations (so this is also the case for order 1 prices).[5]

Example. Let the set of agents be $N = \{1, 2\}$ and the set of items be $M = \{a, b\}$. The agents both have unit-demand valuations, given by the following table.

	a	b
1	1	1
2	2	3

The efficient allocation gives item a to agent 1, and item b to agent 2, for a total value of 4. Recall that the payoff vector where the seller extracts all the surplus is in the core. In this case, $\pi_0 = 4$, $\pi_1 = 0$, and $\pi_2 = 0$ is in the core. Assume there are anonymous CE prices that yield these core payoffs. We must have $p(a) = 1$ and $p(b) = 3$ to ensure the agents get payoffs of 0 from the allocation. But at these prices, agent 2 prefers a to b, so they cannot be CE prices, which gives us a contradiction. Unit-demand valuations satisfy the substitutes condition, and they are also submodular, and hence subadditive. The example therefore shows that in general, order 3 prices are required to price the core with these valuations classes. These conclusions are summarized in Table 3.2.

[5]This may seem to contradict the results of Shapley and Shubik [101]. Note, however, that the pattern of ownership in their model is different: each item is owned by a distinct seller. Hence no seller can extract all the surplus. In our model, there exists a core outcome where the lone seller extracts all the surplus, and the example shows that this outcome cannot be priced using non-anonymous prices.

	CE = Core	VPP ∈ CE	References
Linear	1	1	Immediate
Unit-Demand	3	1	B&O [13], Leonard [62], this work
Substitutes	3	3	B&O [13]
Single-minded	2	×	This work, A&M [7]
Supermodular	2	×	This work, A&M [7]
Superadditive	2	×	This work, A&M [7]
Submodular	3	×	B&O [13], A&M [7]
Subadditive	3	×	B&O [13], A&M [7]
General	3	×	B&O [13], A&M [7]

Table 3.2: This table lists, for each class of valuations, the lowest order of prices that ensures that the core can be priced, and that the Vickrey payoff point (VPP) can be priced.

3.5 Relation to Vickrey payoffs

We have seen how core payoffs can be obtained in competitive equilibrium. It is now natural to ask: when are Vickrey payoffs competitive? Theorem 8 shows that for Vickrey payoffs to arise in a competitive equilibrium, the Vickrey payoff point must lie in the core. Hence, by Theorems 2 and 3, Vickrey payoffs cannot necessarily arise in competitive equilibrium if the agents have general, submodular/supermodular, subadditive/superadditive, or single-minded valuations.

As for additive and unit-demand valuations, Leonard [62] shows that the Vickrey payoff point can be priced with order 1 prices in these cases. On the other hand, when agents have substitutes valuations, Bikhchandani and Ostroy [13] give an example that shows order 3 prices might be required to implement Vickrey payoffs. These conclusions are summarized in Table 3.2. The case of unit-demand valuations shows that whether the core can be priced and whether the Vickrey payoff point can be priced, for a given order of prices, are separate questions.

Since Vickrey payoffs cannot be achieved in competitive equilibrium for several interesting classes of valuations, we might ask: when can Vickrey payments be derived from CE prices? This leads us to the concept of a *universal competitive equilibrium*(UCE), introduced by Mishra and Parkes [71]. Prices p are *UCE prices* if they support any efficient allocation of items among agents M, and also support any efficient allocation among agents $N - i$ for all $i \in N$. An efficient allocation R together with UCE prices p form a *universal competitive equilibrium.*

Theorem 10 [71] *Let (R, p) be a universal competitive equilibrium, and let R' be any revenue-maximizing allocation of the items among agents $N - i$. Then the Vickrey payment to agent i is*

$$\hat{q}_i = \sum_{j \in N-i} p_j(R'_j) - \sum_{j \in N-i} p_j(R_j). \tag{3.10}$$

Proof. Fix some $i \in N$. Let R' be an efficient allocation of the items among agents $N - i$. By definition p supports both R and R'. Hence for any $j \in N$ we have

$$v_j(R_j) - p_j(R_j) = v_j(R'_j) - p_j(R'_j). \tag{3.11}$$

Also, note that R' maximizes the seller's revenue among all allocations to agents $N - i$. We then have

$$\begin{aligned}
\hat{q}_i &= \sum_{j \in N-i} v_j(R'_j) - \sum_{j \in N-i} v_j(R_j) \\
&= \sum_{j \in N-i} \left[v_j(R'_j) - p_j(R'_j)\right] + \sum_{j \in N-i} p_j(R'_j) - \sum_{j \in N-i} \left[v_j(R_j) - p_j(R_j)\right] - \sum_{j \in N-i} p_j(R_j) \\
&= \sum_{j \in N-i} p_j(R'_j) - \sum_{j \in N-i} p_j(R_j)
\end{aligned}$$

where the first equality is (2.3), the definition of the Vickrey payment, and the third follows by cancelling terms according to (3.11). □

Theorem 10 shows that a universal competitive equilibrium contains sufficient information to construct any agent's Vickrey payment: note that the first term in (3.10) can be determined simply with knowledge of p, whereas the second term can obviously be determined with knowledge of R and p.

3.6 Communication

Theorem 4 shows that knowledge of a competitive equilibrium (R, p) is enough to determine an efficient allocation: simply take R itself. Theorem 10 shows that knowledge of a universal competitive equilibrium is enough to determine an efficient allocation together with corresponding Vickrey payments. We now examines the converse of these statements. We will see that any communication protocol that finds an efficient allocation, or an efficient allocation with Vickrey payments, must necessarily identify a competitive or universal competitive equilibrium, respectively.

A *nondeterministic communication protocol* is a triple $\Pi = (\mathcal{M}, \mu, f)$, where $\mathcal{M}$ is a message set, $\mu : \mathcal{V}^N \twoheadrightarrow \mathcal{M}$ is a message correspondence, $f : \mathcal{M} \twoheadrightarrow \mathcal{O}$ is the outcome function, and the message correspondence μ has the following properties:

- *Existence*: $\mu(v) \neq \emptyset$ for all $v \in \mathcal{V}^N$.
- *Privacy preservation*: $\mu(v) = \bigcap_i \mu_i(v_i)$ for all $v \in \mathcal{V}$, where $\mu_i : \mathcal{V} \longrightarrow M$ for all $i \in N$.

Protocol Π *realizes* choice rule $F : \mathcal{V}^N \twoheadrightarrow \mathcal{O}$ if $h(\mu(v)) \subseteq F(v)$ for all $v \in \mathcal{V}$. These definitions can be interpreted as follows. In a nondeterministic communication protocol that realizes a choice F, the fact that the agents all accept a message m

(where i accepts m if $m \in \mu_i(v_i)$) verifies that outcome $h(m)$ is correctly chosen, because $h(m) \in h(\mu(v)) \subseteq F(v)$. (The set of outcomes here may vary according to the rule; it does not need to be precisely a an allocation together with payments as in Chapter 2. It may be simply an allocation, an allocation and payments, or an allocation and prices, for instance.)

The *communication requirement* of a protocol is the worst-case encoding size of a message in bits, i.e. $\log |\mathcal{M}|$, if the message space is discrete. If the message space is continous, the communication requirement is defined as the dimension of the space. The communication complexity of a choice rule is the smallest communication requirement over all protocols realizing the rule.

Consider the *efficient rule*, which maps valuations profiles v into the corresponding set of efficient allocations. Also, consider the *competitive equilibrium rule*, which maps profiles v into the corresponding set of competitive equilibria (R, p). The following fundamental result by Nisan and Segal [76] relates the communication complexities of these rules.[6]

Theorem 11 [76] *Communication protocol* $\Pi = (\mathcal{M}, \mu, f)$ *realizes the efficient rule if and only if there is an assignment* $p : \mathcal{M} \to \mathbf{R}^{N \times 2^M}$ *of prices to messages such that protocol* $(\mathcal{M}, \mu, (f, p))$ *realizes the competitive equilibrium rule.*

Proof. The "sufficient" direction follows from Theorem 4. For the "necessary" direction, suppose protocol $(\mathcal{M}, \mu, f)$ realizes the efficient rule. For each $m \in \mathcal{M}$, let

[6]Parkes [81] provides a similar result in a slightly less general model with a restricted communication language.

$R = f(m)$ and define prices

$$p_i(S) = \sup_{v_i \in \mu_i^{-1}(m)} [v_i(S) - v_i(R_i)]$$

for all $S \subseteq M$ and $i \in N$. Note in particular that $p_i(R_i) = 0$ for all $i \in N$. By construction, the bundles in R maximize agent i's utility for any $v_i \in \mu_i^{-1}(m)$. Also for each $R' \in \Gamma(N)$ we have

$$\begin{aligned} \sum_{i \in N} [p_i(R'_i) - p_i(R_i)] &= \sum_{i \in N} \sup_{v_i \in \mu_i^{-1}(m)} [v_i(R'_i) - v_i(R_i)] \\ &= \sup_{v \in \mu^{-1}(m)} \sum_{i \in N} [v_i(R'_i) - v_i(R_i)] \\ &\leq 0, \end{aligned}$$

where the second equality follows from privacy preservation, and the inequality follows from the fact that R_i is efficient for any $v \in \mu^{-1}(m)$, because Π implements the efficient rule. So R also maximizes the seller's revenue at prices p, and hence (R, p) is a competitive equilibrium, for every $v \in \mu^{-1}(m)$. □

Consider now the *VCG rule* that maps profiles v into pairs (R, q), where R is an efficient allocation with respect to v and q the corresponding vector of Vickrey payments. Recall that the Vickrey payment from agent i corresponding to R is

$$\max_{R' \in \Gamma(N-i)} \sum_{j \in N-i} v_j(R'_j) - \sum_{j \in N-i} v_j(R_j).$$

We would like to compare this with the *universal competitive equilibrium rule* that, as the name implies, maps valuation profiles v into the corresponding set of universal competitive equilibria (R, p). Recall that in a universal competitive equilbirium, prices p not only support R but also any efficient allocation among agents $N - i$, for any $i \in N$. The following theorem is the analog of Theorem 11 for this pair of rules.

Theorem 12 *Communication protocol* $\Pi = (\mathcal{M}, \mu, (f, q))$ *realizes the VCG rule if and only if there is an assignment* $p : \mathcal{M} \to \mathbf{R}^{N \times 2^M}$ *of prices to messages such that protocol* $(\mathcal{M}, \mu, (f, p))$ *realizes the universal competitive equilibrium rule.*

Proof. The "sufficient" direction follows from Theorem 10. For the "necessary" direction, suppose protocol $(\mathcal{M}, \mu, f)$ realizes the VCG rule. For each $m \in \mathcal{M}$, let $R = f(m)$ and define prices

$$p_i(S) = \sup_{v_i \in \mu_i^{-1}(m)} [v_i(S) - v_i(R_i)]$$

for all $S \subseteq M$ and $i \in N$. Note in particular that $p_i(R_i) = 0$ for all $i \in N$. The proof of Theorem 11 shows that prices p support R for every $v \in \mu^{-1}(m)$.

Let $v \in \mu^{-1}(m)$. Fix $i \in N$ and let R' be an efficient allocation among agents $N - i$ with respect to v_{-i}. Assume there is an agent $j \neq i$ such that R'_j does not maximize its utility at prices p. Since R_j does maximize agent j's utility at prices p, as argued above, we have

$$\begin{aligned} & v_j(R'_j) - p_j(R'_j) < v_j(R_j) - p_j(R_j) \\ \Rightarrow \quad & v_j(R'_j) - v_j(R_j) < p_j(R'_j) \\ \Rightarrow \quad & v_j(R'_j) - v_j(R_j) < \sup_{v'_j \in \mu_j^{-1}(m)} [v'_j(R'_j) - v'_j(R_j)]. \end{aligned}$$

Let $s = \sup_{v_j \in \mu_j^{-1}(m)} [v_j(R'_j) - v_j(R_j)]$. By the last inequality above there is some $\epsilon > 0$ such that $v_j(R'_j) - v_j(R_j) = s - \epsilon$. By the definition of a supremum, there is some $v'_j \in \mu_j^{-1}(m)$ such that $v'_j(R'_j) - v'_j(R_j) > s - \epsilon = v_j(R'_j) - v_j(R_j)$. Adding

$\sum_{k\in N\setminus\{i,j\}}[v_k(R'_j)-v_k(R_j)]$ to both sides of this inequality and rearranging, we obtain

$$\begin{aligned}&\sum_{k\in N-i} v_k(R'_j) - \sum_{k\in N-i} v_k(R_j)\\ <\ &\left[\sum_{k\in N\setminus\{i,j\}} v_k(R'_j) + v'_j(R'_j)\right] - \left[\sum_{k\in N\setminus\{i,j\}} v_k(R_j) + v'_j(R_j)\right].\end{aligned}$$

The left-hand side is agent i's Vickrey payment with respect to R when the valuation profile is v. The right-hand side is at most agent i's Vickrey payment with respect to R when the valuation profile is (v'_j, v_{-j}). So the Vickrey payment from i differs under these two profiles. But note that $v \in \mu^{-1}(m)$ and $(v'_j, v_{-j}) \in \mu_j^{-1}(m) \times \mu_{-j}^{-1}(m) = \mu^{-1}(m)$, the latter by privacy preservation. So the Vickrey payment from i must in fact be identically $q_i(m)$ for these two profiles, from the fact that Π realizes the VCG rule. This gives us a contradiction, and we see that R'_j in fact maximizes agent j's utility at prices p.

We have just shown that for $v \in \mu^{-1}(m)$ and R' efficient for agents $N-i$ with respect to v, we have for each $j \in N-i$

$$\begin{aligned}&v_j(R'_j) - p_j(R'_j) = v_j(R_j) - p_j(R_j)\\ \Rightarrow\ &p_j(R'_j) = v_j(R'_j) - v_j(R_j).\end{aligned}$$

Summing this over all $j \in N-i$, we have

$$\sum_{j\in N-i} p_j(R'_j) = \sum_{j\in N-i} [v_j(R'_j) - v_j(R_j)].$$

Because R' is efficient for agents $N-i$, for any $R'' \in \Gamma(N-i)$ we then have

$$\sum_{j\in N-i} p_j(R'_j) \geq \sum_{j\in N-i} [v_j(R''_j) - v_j(R_j)].$$

Finally, because this holds for any $v \in \mu^{-1}(m)$, we have

$$\begin{aligned}\sum_{j\in N-i} p_j(R'_j) &\geq \sup_{v\in\mu^{-1}(m)} \sum_{j\in N-i} [v_j(R''_j) - v_j(R_j)] \\ &= \sum_{j\in N-i} \sup_{v_j\in\mu_j^{-1}(m)} [v_j(R''_j) - v_j(R_j)] \\ &= \sum_{j\in N-i} p_j(R''_j),\end{aligned}$$

where the first equality follows from privacy preservation. As this holds for any $R'' \in \Gamma(N-i)$, we see that R'_j maximizes the revenue to the seller among allocations to agents $N-i$. This confirms that p are UCE prices for any $v \in \mu^{-1}(m)$. □

Figure 3.1 gives the intuition behind this result for the case with two agents. Given a message m, the figure shows all valuations that are consistent with m for agent 1 (here only v_1) and for agent 2 (v'_2 and v''_2). The valuations are normalized so that the efficient allocation R has value 0. The conditions for a CE require that all of agent 1's valuations consistent with m be above all of agent 2's valuations consistent with w. To construct valid CE prices, we take the envelopes of agent 1 and 2's valuations (lower and upper, respectively). Since agent 2's valuations are all consistent with the same Vickrey payments, they all peak at the same level. This ensures that the envelope of agent 2's valuations touches these peaks, so that the constructed CE prices also satisfy the UCE constraints.

The notions of CE and UCE prices are clearly closely related. The proof of the next result shows how UCE prices can be constructed from CE prices from the main economy and the marginal economies.

Theorem 13 *If there exist competitive equilibrium prices of dimension d for all profiles of valuations drawn from $\mathcal{V}^N$, then there exist universal competitive equilibrium*

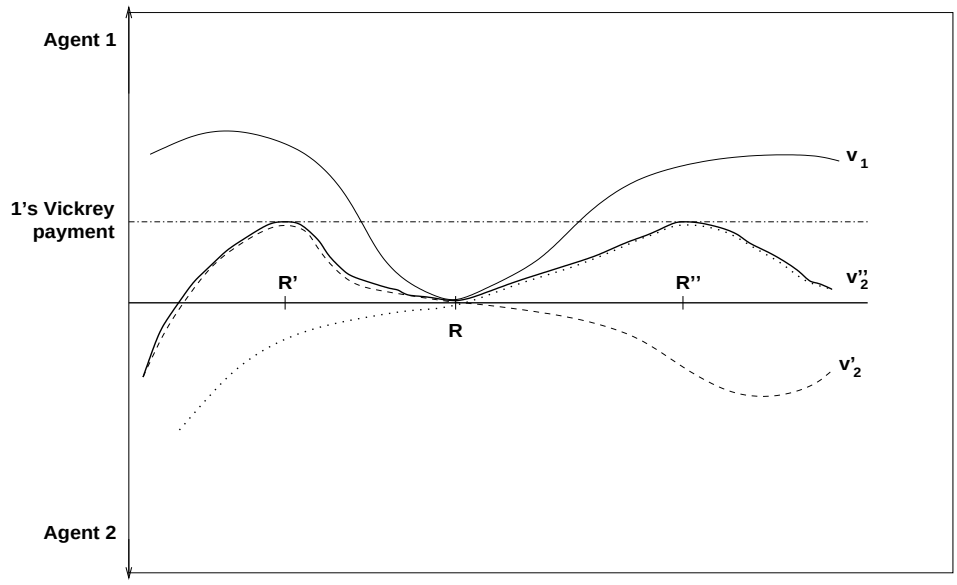


Figure 3.1: Constructing UCE prices.

prices of dimension $(n+1)d + n^2$ *for all profiles of valuations drawn from* $\mathcal{V}^N$.

Proof. Let $(S_i^{-k})_{i \in N-k}$ be an efficient allocation among agents $N - k$ (where by convention all agents are present when $k = 0$), and let p_i^{-k} be supporting CE prices, for $k = 0, \ldots, n$. By assumption each of these supporting price vectors has dimension at most d. Consider the following prices.

$$\bar{p}_i(S) = \min_{k=0,\ldots,n} \left\{ p_i^{-k}(S) + \pi_i^{-k} \right\} \tag{3.12}$$

First note that $\bar{p}_i(S_i^{-k}) = p_i^{-k}(S_i^{-k}) + \pi_i^{-k}$ for $k = 0, \ldots, n$. Otherwise there must be an $l \neq k$ such that

$$\begin{aligned} & p_i^{-l}(S_i^{-k}) + \pi_i^{-l} < p_i^{-k}(S_i^{-k}) + \pi_i^{-k} \\ \Rightarrow \quad & \pi_i^{-l} < v_i(S_i^{-k}) - p_i^{-l}(S_i^{-k}) \end{aligned}$$

which is a contradiction, because π_i^{-l} represents the maximum utility that i can attain

at prices p_i^{-l}. We thus find that

$$\begin{aligned} v_i(S_i^{-k}) - \bar{p}_i(S_i^{-k}) &= v_i(S_i^{-k}) - p_i(S_i^{-k}) - \pi_i^{-k} \\ &= 0 \end{aligned}$$

for $k = 0, \ldots, n$. Now let S_i be an arbitrary bundle, and let l be the index at which the minimum is reached for S_i on the right-hand side of (3.12). We have

$$\begin{aligned} v_i(S_i) - \bar{p}_i(S_i) &= v_i(S_i) - p_i^{-l}(S_i) - \pi_i^{-l} \\ &\leq v_i(S_i^{-l}) - p_i^{-l}(S_i^{-l}) - \pi_i^{-l} \\ &= 0 \\ &= v_i(S_i^{-k}) - \bar{p}_i(S_i^{-k}) \end{aligned}$$

for k $0, \ldots, n$, where the inequality follows from the fact that S_i^l maximizes agent i's utility at prices p_i^{-l}. Since S_i was arbitrary, S_i^{-k} maximizes agent i's utility at prices $\bar{p}_i$ for $k = 0, \ldots, n$.

Finally, fix $k \in \{0, \ldots, n\}$ and let $S = (S_i)_{i \in N}$ be an arbitrary feasible allocation among agents $N - k$. We have

$$\begin{aligned} \sum_{i \in N-k} \bar{p}_i(S_i) &\leq \sum_{i \in N-k} p_i(S_i) + \sum_{i \in N-k} \pi_i^{-k} \\ &\leq \sum_{i \in N-k} p_i(S_i^{-k}) + \sum_{i \in N-k} \pi_i^{-k} \\ &= \sum_{i \in N-k} \bar{p}_i(S_i^{-k}) \end{aligned}$$

Hence S^{-k} maximizes revenue among allocations to agents $N - k$ at prices $\bar{p}$, for k $0, \ldots, n$. This shows that $\bar{p}$ are universal CE prices. The dimension of prices $\bar{p}$ is $(n+1)d + n^2$, which proves the theorem. $\square$

To verify a universal competitive equilibrium, we simply need to broadcast the efficient allocations for the marginal economies and main economy together with UCE prices. The agents then accept the message if and only if their allocated bundle in each economy maximizes their utility at the given prices. That the allocations maximize the auctioneer's revenue can be checked simply with knowledge of the prices. The allocations vector has dimension n^2, which leads us to the next result.

Theorem 14 *The communication requirements of the VCG rule and the universal competitive equilibrium rule are at most* $(n+1)d + n^2$*, where* $d + n$ *are the communication requirements of the efficient rule.*

Theorem 12 can be used to prove lower bounds on the communication requirements of realizing the VCG rule, just as Nisan and Segal [76] use Theorem 11 to prove lower bounds for the efficient rule. The following result gives an example of this for the case where the items are identical and the agents' valuations exhibit decreasing marginal values.

Proposition 4 *Suppose the items are identical and each agent's valuation exhibits decreasing marginal values. Then an efficient protocol requires a message space of dimension no more than 1, whereas a protocol that verifies the VCG rule has a message space of dimension at least* $\min\{n-1, m\}$.

Proof. Let agents $1, \ldots, n-1$ have unit-demand valuations, so that the marginal value of any item beyond the first is 0. We can describe these valuations by the marginal value of the first unit, denoted v_i for agent i. Agent n has an additive valuation, and the marginal value of any item is v_n. Consider the set of valuation

profiles

$$\mathcal{F} = \{(v_i)_{i \in N} \,|v_1 > v_2 \ldots > v_{n-1}, v_n > v_1\}\,.$$

For profiles drawn from this set, it is efficient to give all items to agent n. However, if agent n is removed, it is efficient to give one item to each of agent $1, \ldots, k$ and discard the rest, where $k = \min\{n-1, m\}$.

A single real number p describing the marginal price for any item is sufficient to support the efficient allocation: taking $v_1 \leq p \leq v_n$ yields a CE. By Theorem 11, the (nondeterministic) communication requirements of the efficient rule is therefore at most 1.

We claim that any two profiles $v, v' \in \mathcal{F}$ that differ in any of the first k components must lead to different UCE prices. To see this, note that by our arguments above, UCE prices must simultaneously make both 0 and 1 units utility-maximizing for the first k agents. Letting i be one of these agents, we must then have $v_i \cdot 1 - p_i(1) = v_i \cdot 0 - p_i(0) = 0$, and hence $p_i(1) = v_i$. So the vector of UCE prices must coincide with the vector of valuations in the first k components. By Theorem 12, we then see that

$$(n-2, n-1) \times \ldots \times (n-k+1, n-k) \times \{n-k+1\} \times \ldots \times \{1\} \times \{n-1\}$$

is a fooling set of dimension k for the VCG rule, i.e. any two profiles from this set require different messages to correctly realize the VCG rule, because they must lead to distinct UCE prices. To verify the VCG rule, the message space must therefore have dimension at least k. □

Although the increase in communication required by the VCG rule in this case is only polynomial, this is the first result to quantify the strict increase in commu-

nication implied by verifying Vickrey payments together with the efficient outcome.[7] Recall that the VCG mechanism is the mechanism of choice for aligning incentives in our context, because of its revenue properties. Proposition 4 reveals that aligning incentives in this way may, however, come at an extra communication cost.

Fadel and Segal [36] consider the increase in communication implied by incentive-compatibility in general, and they also consider deterministic protocols. They give a simple example in which computing an efficient outcome requires 3 bits of communications, while computing transfers to induce truthfulness necessarily requires a protocol that uses more than 3 bits. Proposition 4 is able to provide a stronger separation because it focuses on the VCG mechanism rather than incentive-compatible mechanisms more generally.

[7]Note that this is a worst-case result. In some instances (e.g. with unit-demand valuations), it may be the case that no added communication is required to verify Vickrey payments.

Chapter 4

Representations

In a combinatorial auction, a bidder would need $2^m - 1$ numbers to exhaustively describe its valuation function. This is impractical for even moderately-sized m such as 15, so bidders need to communicate their values using a more succinct encoding. This motivates the use and study of *bidding languages* to represent valuations. Formally, a language consists of syntax and semantics. The syntax specifies which sequences of symbols are valid elements of the language; the semantics specify how to interpret sequences as values for bundles.

The formal study of bidding languages for combinatorial auctions was initiated by Nisan [74]. Nisan's main focus is on the expressiveness of various languages. He examines which languages can succinctly represent natural valuation classes such as the additive and single-minded valuations. He argues that bidding languages should strike a balance between two goals:

1. There needs to be effective ways to solve the allocation problem when valuations are expressed in the chosen language.

2. Common valuations should be easy to express in the chosen language.

To address the first point, in this chapter we provide a general description of the properties of bidding languages that make them suitable for bidding and winner determination in our context. This then motivates analogous requirements for languages that are used for pricing rather than bidding. Such "pricing languages" have not been examined in the academic literature because expressive bidding has only been developed for single-shot auctions to date, whereas iterative designs have been restricted to linear or XOR prices, as we saw in Section 1.2.2. Expressive pricing is necessary to guarantee efficiency, because Nisan and Segal [76] have shown that prices must equal valuations (up to a constant) in the worst case, for any efficient combinatorial auction faced with general valuations.

As for the second point, Nisan explains that "easy to express" means both that common valuations should have succinct encodings, and also that it should be easy for bidders to express their valuations in the chosen language, either directly or via agent programs. The encoding size of various valuations in various languages is the focus of his study. On the other hand, he does not address how bidders should go about encoding their valuations or how "ease of encoding" should be measured. One way to view the translation problem is through the lens of computational learning theory [49]. A bidder can be construed as a black box that can respond to certain well-defined types of queries about their valuations, and we then need an algorithm that can reconstruct the bidder's valuation in the language of choice given access to this black box. The complexity of the translation problem is then the query complexity of the learning problem, i.e. the number of different types of queries that must be

performed. In this chapter, we propose learning algorithms for several common and novel languages.

Section 4.1 describes the typical query functionality required of languages in the context of one-shot and iterative auctions, introducing value and demand queries. Section 4.2 introduces several representation classes in terms of their syntax and semantics. Some are drawn from the literature on CAs, some are novel. For each class we discuss the complexity of responding to value and demand queries, and whether the representation is suitable for bidding and pricing. Section 4.3 introduces the query learning model that provides a measure of performance for algorithms that encode valuations into specific representations. Section 4.4 then goes on to provide learning algorithms for several representation classes. Section 4.5 concludes with a discussion of other bidding languages and learning algorithms in the literature that were not covered earlier in the chapter.

4.1 Functionality

4.1.1 Value Queries and Bidding

The most basic functionality required of a language is the value query.

Value query. A bundle S is input and the value $v(S)$ of the bundle is returned, according to the valuation represented by the language.

Ideally, an algorithm to evaluate value queries should be provided with each language. Mathematical programming methods find widespread use in the practice of combinatorial auctions, so one way to ensure a practical algorithm exists is to specify the

value query problem as a mathematical program. Typically this will be an integer or mixed-integer program. A wealth of effective algorithms and heuristics may then be applied to the problem [73].

We say that a language is *suitable for bidding* if the problem of evaluating a value query can be formulated as a mixed-integer program whose size is polynomial in t, the size of the given representation, and whose objective is a *maximization*. Specifically, let $y = (y_j)_{j \in M}$ be a 0-1 indicator vector for a bundle. There should be matrices A and B, as well as vectors b and c such that the value of the program

$$\begin{aligned} \max_{x} \quad & c'x \\ \text{s.t.} \quad & Ax + By \leq b \end{aligned}$$

gives the value of the bundle corresponding to y. Here y is a constant vector of size m and x is a vector of auxiliary variables whose size is polynomial in t. The size of b should also be polynomial in t, and the sizes of A, B, and c should agree with the other vectors. Some components of x may be restricted to be integer.

Besides providing a way to evaluate value queries, the MIP formulation also makes it simple to incorporate the represented valuation into the auctioneer's winner-determination problem—hence the phrasing "suitable for bidding." Given (A_i, B_i, b_i, c_i) for each agent $i \in N$, the winner-determination problem can be formulated as

$$\begin{aligned} \max_{x,y} \quad & \sum_{i \in N} c_i' x_i \\ \text{s.t.} \quad & A_i x_i + B_i y_i \leq b_i \quad (i \in N) && (4.1) \\ & \sum_{i \in N} y_{ij} \leq 1 \qquad (j \in M) && (4.2) \end{aligned}$$

Here y_i is now the indicator vector for the bundle allocated to agent i, and it becomes a choice variable. The entries of the vector must lie in $\{0, 1\}$. Constraints (4.2) ensure that the computed allocation is feasible. The remaining MIP logic is separable into the formulations corresponding to the instances of the agents' languages.

Bid shifts

It can often be useful to shift valuations down by a constant α, so that we represent the valuation $\max\{v_i - \alpha, 0\}$ rather than v_i. This is particularly useful when computing bidder-optimal core payoffs. We adapt the MIP formulation of a bidding language to introduce a downward shift of α as follows.

$$\begin{aligned} \max_{x} \quad & c'x - \alpha z \\ \text{s.t.} \quad & Ax + By \leq b \\ & z \geq y_j \qquad (j \in M) \\ & z \in \{0, 1\} \end{aligned}$$

This does not quite represent the shifted valuation, because some bundles will have negative values. For our purposes, though, this is irrelevant, because an instance of a bidding language is always used in an allocation context. A winner-determination algorithm will always allocate the empty set rather than a bundle with negative value: note that the empty set has value 0 by our constraints on the auxiliary z variable.

4.1.2 Demand Queries and Pricing

Value queries and bidding languages are relevant in the context of one-shot auctions. In iterative designs, it is often useful that agents respond to prices.

Demand query. A bundle S together with prices p are input. If S maximizes the agent's utility at prices p, it replies YES; otherwise some other utility-maximizing bundle is returned.

We make no assumptions on the returned bundle if the agent does not reply YES; for example, it may be chosen adversarially.

The prices themselves must of course also be represented using some language, and there needs to be an effective algorithm for evaluating a demand query. This can be tricky, because the algorithm must incorporate information from the agent's valuation together with the price information. Again, mathematical programming methods provide an elegant way to formulate the problem. We say that a language is *suitable for pricing* if the problem of evaluating a value query[1] can be formulated as a mixed-integer program whose size is polynomial in t, the size of the given representation, and whose objective is a *minimization*. The structure of the formulation parallels the structure for bidding: letting $(\bar{A}, \bar{B}, \bar{b}, \bar{c})$ be the corresponding matrices and vectors, the program is

$$\begin{aligned} \min_{x} \quad & \bar{c}'\bar{x} \\ \text{s.t.} \quad & \bar{A}\bar{x} + \bar{B}y \geq \bar{b} \end{aligned}$$

To see how this structure allows one to evaluate demand queries, suppose the valuation is represented by (A, B, b, c) in a language suitable for bidding, and that the prices are represented by $(\bar{A}, \bar{B}, \bar{b}, \bar{c})$ in a language suitable for pricing. The MIP

[1] A more intuitive name in this case would perhaps be 'price query' rather than 'value query,' but we use the latter to underline the equivalence between bidding and pricing language functionality in this case. Some languages can function both as bidding and pricing languages.

to evaluate a demand query is then

$$\begin{aligned} \max_{x,y} \quad & c'x - \bar{c}'\bar{x} \\ \text{s.t.} \quad & Ax + By \leq b \\ & \bar{A}\bar{x} + \bar{B}y \geq \bar{b} \end{aligned}$$

Note that in this case the y vector is a choice variable.

Price shifts

It can often be useful to shift prices down by a constant. For instance, recall from Section 3.4 that if $v = (v_i)_{i \in N}$ are the bidder valuations and $\pi = (\pi_i)_{i \in N}$ their core payoffs, then the prices defined by

$$p_i(S) = \max\{v_i(S) - \pi_i, 0\}$$

are CE prices that yields core payoffs π. We can adapt the MIP formulation of a pricing language to introduce a price shift α as follows.

$$\begin{aligned} \min_{x,z} \quad & z \\ \text{s.t.} \quad & \bar{A}\bar{x} + \bar{B}y \geq \bar{b} \\ & z \geq \bar{c}'\bar{x} - \alpha \\ & z \geq 0 \end{aligned}$$

If we can represent each agent's valuation in a pricing language, it is then possible to quote order 3 CE prices that give the agents core payoffs π.

4.2 Languages

We now describe several languages in terms of how they can serve as a bidding or pricing language for a single agent.

4.2.1 OR*

The OR* language was introduced by Fujishima et al. [39] and studied in depth by Nisan [74], who showed that it was as succinct for expressing various interesting kinds of valuations as other common languages. We describe it here because it generalizes the OR and XOR languages below. An *atomic bid* is a bundle-value pair (S, v). The bundle may consist of items from M, and also of "dummy items" drawn from a set M^+. The dummy items are agent-specific. The syntax of an OR* bid is simply a list of atomic bids, where in each bid $S \subseteq M \cup M^+$. To evaluate the value of a bundle $T \subseteq M$, we find the maximum packing of bundles from atomic bids into $T \cup M^+$.

Example. Suppose we have items A and B. Consider the valuation that is 1 if at least one item is obtained, and 0 otherwise. We introduce dummy item a, and the OR* representation is then

$$(Aa, 1); (Ba, 1)$$

The dummy item ensures that the value of AB will be subadditive.

The OR* language is suitable for bidding. The integer program that computes the answer to a value query is as follows. Let $\mathcal{B}$ be the set of bundles that appear in

atomic bids. There is a variable x_S for each $S \in \mathcal{B}$.

$$\begin{aligned} \max_{x_b} \quad & \sum_{S \in \mathcal{B}} v_S x_S \\ \text{s.t.} \quad & \sum_{S \ni j} x_S \leq y_j \quad (j \in M) && (4.3) \\ & \sum_{S \ni j} x_S \leq 1 \quad (j \in M^+) && (4.4) \\ & x_S \in \{0,1\} \quad (S \in \mathcal{B}) \end{aligned}$$

We know of no way to formulate OR* semantics as a minimization, so OR* is currently unsuitable for pricing.[2] In this instance, computing the solution to the integer program is NP-hard by reduction from weighted set packing [89], so enumerative approaches such as branch-and-bound are required to obtain the solution. There is a significant amount of research into effective algorithms and heuristics for the kind of packing problem represented by OR* instances [2, 39, 74, 93].

OR* can represent any valuation from the class of general valuations. Nisan [74] shows that OR* is as succinct for representing such valuations as the OR and XOR languages given below.

4.2.2 OR

The OR language is simply the OR* language with no dummy items. It was introduced by Sandholm [93]. Since it is a special case of OR*, OR is also suitable for bidding. The integer program that gives the answer to a value query is identical

[2]This is because we require MIP formulations for value queries to be polynomial in the size of the price representation. It would be possible to formulate a MIP with an exponential number of constraints and use a constraint-generation approach, but this would be very inefficient in and of itself, and would be impractical given the large number of value queries certain elicitation schemes could require.

to the one for OR*, except that we take $M^+ = \emptyset$ so that there are no dummy items. Solving the integer program still remains NP-hard, for the same reasons. Again, we know of no way to formulate OR logic as a minimization MIP, so OR is currently unsuitable as a pricing language.

OR cannot represent all general valuations; in particular, it cannot represent valuations that are strictly subadditive. However, it is still relevant in our context because it is well-suited to complementarities: it is not hard to show that the class of valuations that can be represented by the OR language is exactly the class of superadditive valuations [74].

4.2.3 XOR

The XOR language, also introduced by Sandholm [93], is the OR* language with a unique dummy item that is shared among all atomic bids. This means that only one atomic bid can be selected. Therefore the problem of computing the value of a bundle is no longer NP-hard. In fact, we simply need to scan all atomic bundles that are subsets of the given bundle and take the maximum value of these, so a value query can be computed in linear time.

The integer program for value queries is identical to that for OR*, except that there is always just a single dummy item shared by all atomic bids. Constraints (4.4) therefore consist of the single constraint

$$\sum_{b \in B} x_b \leq 1.$$

The XOR value query logic can also be captured by a minimization MIP, so the XOR language is also suitable as a pricing language. Let $\mathcal{B}$ be the set of bundles that

appear in atomic bids. There is a variable x_S for each $S \in \mathcal{B}$.

$$\begin{aligned} \min_{x_S, z} \quad & z \\ \text{s.t.} \quad & x_S - 1 \geq \sum_{j \in S} y_j - |S| \quad (S \in \mathcal{B}) \\ & z \geq v_S x_S \quad (S \in \mathcal{B}) \\ & x_S \in \{0, 1\} \quad (S \in \mathcal{B}) \\ & z \geq 0 \end{aligned}$$

XOR can represent exactly the class of general valuations. (XOR cannot represent other valuations that do not satisfy free-disposal.) It is useful when a bidder has distinct values for only a few bundles. In other cases, such as the additive valuation, the XOR representation needs to be exponential in size [74].

4.2.4 Polynomials

A polynomial is a sum of terms, where a term is a product of variables, e.g. $y_1 y_3 y_4$, times some coefficient drawn from the real numbers. As usual, the y_j here are 0-1 indicator variables for the items. Lahaie and Parkes [59] first proposed polynomials to represent valuation functions. Each valuation from the class of general valuations has a unique representation as a polynomial [99]. (However, polynomials can also represent valuations that do not satisfy free-disposal.)

Example. Suppose there are two items A and B, and that the valuation is $v(A) = 1$, $v(B) = 2$, and $v(AB) = 2$. Then the polynomial representation is

$$(A, 1); (B, 2); (AB, -1)$$

Here we have described terms as atomic bids. An atomic bid is selected if it is a subset of the given bundle. The coefficients of all selected atomic bids are added to give the bundle's value.

Polynomials are suitable as a bidding and pricing language. The following integer program selects exactly those terms that are subsets of the given bundle. Let $\mathcal{B}$ be the set of bundles that appear in the terms (atomic bids) of the representation. There is a variable x_S for each $S \in \mathcal{B}$.

$$\sum_{S \ni j} x_S \leq y_j \qquad (j \in M) \tag{4.5}$$

$$x_S - 1 \geq |S| - \sum_{j \in S} y_j \quad (S \in \mathcal{B}) \tag{4.6}$$

Constraints (4.5) ensures that a term cannot be selected unless all its items are present in the given bundle S'. Constraints (4.6) ensure that a term is selected if all its items appear in S'. The unique solution is therefore exactly the set of all terms that are subset of S'.

To complete the integer program formulation, we append the objective $\sum_{S \in \mathcal{B}} v_S x_S$, and we use a maximization or minimization depending on whether the representation is used for bidding or pricing, respectively. Since the solution to the constraints is unique, the direction of the objective is in fact irrelevant.

To get an idea of the succinctness of polynomials as a bidding language, consider the *single-item* valuations presented by Nisan [74]. In the single-item valuation, all bundles have value 1, except $\emptyset$ which has value 0 (i.e. the agent is satisfied as soon as it has acquired a single item). It is not hard to show that the single-item valuation requires polynomials of size $2^m - 1$, while polynomials of size m suffice for the additive

valuation. This is in contrast to the XOR language, where the reverse situation holds. Polynomials are thus appropriate for valuations that are "mostly additive," with a few substitutabilities and complementarities that can be introduced by adjusting coefficients.

4.2.5 Pseudo-additive

We introduce here a new language suitable for bidding and pricing which we call *pseudo-additive representations*. This language will prove useful for describing prices in the auction scheme developed in Chapter 6. This language can also be described in terms of atomic bids, but, as with polynomials, the constraints that dictate which bids should be selected are very different from the constraints that arise with OR*, OR, and XOR. The language can describe any general valuation (and in fact even valuations that do not satisfy free-disposal), and can succinctly express additive valuations.

To describe the language, we first need the notion of a *pattern*. A pattern is simply an ordered list of bundles $\mathcal{B}$, with no bundle being repeated. A *decomposition* of a bundle S into bundles from $\mathcal{B}$ is a set of pairwise disjoint bundles from $\mathcal{B}$ that are all contained in S. A pattern implies a decomposition of any bundle into bundles from the pattern according to Algorithm 1. We say that bundle S_1 *clutters* bundle S_2 if $S_1 \cap S_2 \neq \emptyset$ and $S_1 \not\subset S_2$. We write $S_1 \diamond S_2$ to denote that S_1 clutters S_2.

The time required to compute the decomposition of a bundle is quadratic in the size of the pattern in the worst-case. Note that if the pattern contains all singletons $\{j\}$ for $j \in M$, then the resulting decomposition $\{S_1, \ldots, S_k\}$ will satisfy $\bigcup_{l=1}^{k} S_l = S$. Also, if $S \in \mathcal{B}$, then the decomposition of S according to $\mathcal{B}$ is the singleton $\{S\}$.

Input: A pattern $\mathcal{B}$ and a bundle S.

Output: A decomposition of S into bundles from $\mathcal{B}$.

Let $\mathcal{D} := \emptyset$.

foreach $T \in \mathcal{B}$ *such that* $T \subseteq S$ *(in order)* **do**

  if *there is no* $T' \in \mathcal{D}$ *such that* $T' \diamond T$ **then**

    Add T to $\mathcal{D}$.

    Remove all subsets of T from $\mathcal{D}$.

  end

end

Algorithm 1: Decomposing a bundle according to a pattern.

To complete the description of a valuation, we associate a value to each bundle in the pattern, which gives us atomic bids. The value of a bundle is the sum of the values associated with the bundles in its decomposition.

Example. Suppose there are three items A, B, and C, and that the valuation is as follows.

S	A	B	C	AB	AC	BC	ABC
$v(S)$	1	1	1	3	1	2	4

In this case, the pseudo-additive representation is

$$(A, 1); (B, 1); (C, 1); (AB, 3); (AC, 1)$$

The pattern is (A, B, C, AB, AC). The decomposition of bundle ABC according to Algorithm 1 is $\{AB, C\}$, which leads to a value of 4 as required. Note that the order of the bundles in the pattern is crucial. If we switched bundles AB and AC in the order, the decomposition of ABC would become $\{AC, B\}$ and the value of ABC according to the pseudo-additive instance would become 2.

In valuation v, the value of AB is superadditive because its constituent items are complements, whereas the value of AC is subadditive because its constituent items are substitutes. The value of BC is additive, which is why it does not need to be explicitly listed in the pattern. Its decomposition is $\{B, C\}$ which gives a value of 2, in agreement with v.

Like polynomials and unlike XOR, the pseudo-additive language can succinctly represent additive valuations: the pattern simply consists of the items and their individual values, in any order.

The decomposition of a bundle is given as the unique solution to the following set of constraints. Here we have a 0-1 variable x_S for each $S \in \mathcal{B}$, and the usual indicator vector y to denote the bundle whose value is queried. We write $S_1 \leq S_2$ for $S_1, S_2 \in \mathcal{B}$ to denote that S_1 appears before S_2 in the pattern (or $S_1 = S_2$). Let $l(S) = \{S' < S \mid S' \diamond S\}$, the set of bundles that appear before S in the pattern and that clutter S. Let $s(S) = \{S' \geq S \mid S' \supseteq S\}$, the set of bundles that appear after S and that contain S (including S itself).

$$\sum_{S \ni j} x_S \leq y_j \qquad (j \in M) \tag{4.7}$$

$$\sum_{S' \in s(S)} x_{S'} - 1 \geq \sum_{j \in S} y_j - |S| - \sum_{S' \in l(S)} x_{S'} \qquad (S \in \mathcal{B}) \tag{4.8}$$

$$x_S \in \{0, 1\} \qquad (S \in \mathcal{B}) \tag{4.9}$$

Since it is not obvious that this formulation gives the correct solution, we prove this now.

Lemma 4 *Constraints (4.7)–(4.9) have a unique solution, which is the indicator vector of the decomposition according to Algorithm 1 of the bundle indicated by y .*

Proof. Constraints (4.7) ensure that the resulting decomposition will consist of mutually disjoint bundles that are all subsets of T. The term $\sum_{j \in S} y_j - |S|$ in the constraint for S in (4.8) renders the constraint obsolete if not all items from S are present in T: in this case, the left-hand side is necessarily at least -1 whereas the right-hand side is at most -1. If $S \subseteq T$, on the other hand, the term is 0. So in evaluating the solution to the program, we can ignore those constraints (4.8) for which $S \not\subseteq T$, and among those that remain ignore the term $\sum_{j \in S} y_j - |S|$.

For simplicity, we first assume that the pattern $\mathcal{B}$ contains all singletons $\{j\}$ for $j \in M$ listed a the beginning. Note that this does not affect the remaining bundles chosen for any decomposition; however, it ensures that the union of the resulting will be exactly T.

In the base case, the decomposition $\mathcal{D}$ of bundle T consists only of items. Then by constraints (4.8) all items $j \in T$ must have $x_{\{j\}} = 1$ and by constraints (4.7) items $j \notin T$ must have $x_{\{j\}} = 0$. The unique solution to the constraints is then the decomposition of T into its constituent items, which is exactly $\mathcal{D}$.

For the induction step, consider the first non-singleton bundle S in pattern $\mathcal{B}$ that is a subset of T. When Algorithm 1 encounters S, it adds it to the current decomposition. As a result the items in S will appear in the same bundle in the final decomposition, because S clutters any bundle that it not a strict superset or disjoint. We can therefore redefine S as a new "item" and proceed with the algorithm accordingly. Similarly, the right-hand side of constraint S in (4.8) evaluates to 0, because as S is the first non-singleton no previous bundle can clutter it. Hence either it or a superset must be set to 1 in the final solution. Again we can redefine S to

be an item and set to 0 the variables corresponding to bundles that S clutters, and eliminate their corresponding constraints from (4.8). Hence pattern $\mathcal{B}$ now consists of at least one less non-singleton. Eventually, we reach the base case.

When the base case is reached, we have a unique solution as explained. In this solution, $x_S = 1$ if and only if the items in S appear together in $\mathcal{D}$, so the solution indeed corresponds to the decomposition computed by Algorithm 1. □

We complete the formulation by appending the objective $\sum_{S \in \mathcal{B}} v_S x_S$. The direction of the objective is irrelevant since the solution to the constraints is unique. By convention, we use a maximization for bidding and a minimization for pricing.

4.3 Query Learning

The query learning model we consider here is called *exact learning from membership and equivalence queries*, introduced by Angluin [3]. In this model, the learning algorithm's objective is to exactly identify an unknown *target* function $f : X \to Y$ via queries to an *oracle*. The target function is drawn from a function class $\mathcal{C}$ that is known to the algorithm. Typically the domain X is some subset of $\{0,1\}^m$, and the range Y is either $\{0,1\}$ or some subset of the real numbers $\mathcal{R}$. As the algorithm progresses, it constructs a *manifest* hypothesis $\tilde{f}$ which is its current estimate of the target function. Upon termination, the manifest hypothesis of a correct learning algorithm satisfies $\tilde{f}(x) = f(x)$ for all $x \in X$.

It is important to specify the representation that will be used to encode the manifest hypothesis, since the learning algorithm must be designed with respect to some representation. Let $size(f)$ be the size of the encoding of function f with respect to

the chosen representation. Most representations have a natural measure of encoding size; e.g. the size of a DNF formula is the number of terms in the formula.

Two types of queries are commonly used for exact learning: *value*[3] and *equivalence* queries. Value queries were described in Section 4.1.1.

Equivalence query. The learning algorithm presents its manifest hypothesis $\tilde{f}$. The oracle either replies YES if $\tilde{f} = f$, or returns a *counterexample* x such that $\tilde{f}(x) \neq f(x)$.

An equivalence query is *proper* if $size(\tilde{f}) \leq size(f)$ at the time the query is made.

We are naturally interested in efficient learning algorithms, that run in time polynomial in all the relevant parameters. The following definition is drawn from Kearns and Vazirani [49]:

Definition 1 *The representation class $\mathcal{C}$ is* efficiently exactly learnable from value and equivalence queries *if there is a fixed polynomial $p(\cdot, \cdot)$ and an algorithm L with access to membership and equivalence queries of an oracle such that for any target function $f \in \mathcal{C}$, L outputs in time $p(size(f), m)$ a function $\tilde{f} \in \mathcal{C}$ such that $\tilde{f}(x) = f(x)$ for all instances x.*

Here m is the dimension of the domain. Since the target function must be reconstructed, we also necessarily allow polynomial-time dependence on $size(f)$.

[3]Value queries are sometimes called 'membership queries' in the computational learning theory literature. The reason for this is that the target functions are usually boolean, so instances are either members of the target concept (target function value of 1) or not (target function value of 0).

4.4 Learning Algorithms

The query learning model applies directly to the problem of encoding bidder valuations. In our context, the function class is the valuation class from which a bidder's valuation is drawn. The target function is the bidder's valuation. We will refer to the intermediate estimate of the target valuation as a learning algorithm is run as the *manifest valuation*.

The languages mentioned in Section 4.2 all have natural measures of encoding size: the size of an XOR, OR, or OR* bid is the number of atomic bids; the size of a polynomial is the number of terms; and the size of a pseudo-additive bid is the number of bundles in the pattern.

The question then is whether there exists a learning algorithm that will recover the target valuation in the language of our choice. In the algorithms given below, we allow for value and equivalence queries as in the usual query learning model. We also allow for demand queries, because these are natural in an auction context. Equivalence queries are unnatural in this context, but we will see later in Chapter 5 that they can be replaced with demand queries in iterative schemes. Hence it is still useful to develop algorithms that use equivalence queries.

4.4.1 XOR

Lahaie and Parkes [59] give a learning algorithm for XOR representations that uses value and demand queries. The algorithm is closely related to Angluin's algorithm for monotone DNF [4], which is itself an adaptation of an algorithm by Valiant [104].

Given: A value and demand query oracle for the valuation.

Output: The minimal XOR representation of the target valuation.

Let v be the XOR bid $\emptyset$.

(Throughout, v has XOR semantics.)

repeat

Present $\emptyset$ together with prices v as a demand query.

Record the response S.

Set $T := S$, or $T := \emptyset$ if response was YES.

foreach $j \in T$ **do**

Perform value queries on T and $T - j$.

if *the values are equal* **then**

Set $T := T - j$.

end

end

Let w be the value of T.

Set $v := v \cup (T, w)$.

until *response is* YES

Return v as an XOR representation.

Algorithm 2: Learning algorithm for the XOR language.

Lemma 5 *Algorithm 2 outputs the minimal XOR representation of the target valuation.*

Proof. Assume by induction that all atomic bids in the manifest valuation are also present in the target valuation's minimal XOR representation. This is true initially, when the manifest valuation is set to the empty XOR representation. The manifest

valuation thus lower bounds the true valuation of any bundle, by XOR semantics. If all atomic bids have been discovered, the empty set maximizes utility and YES will be returned upon a demand query, and the algorithm correctly halts. Otherwise, some other bundle is returned, and some subset of this bundle must necessarily appear in some undiscovered atomic bid, by XOR semantics.

Let S be the returned bundle and let $T \subseteq S$ be the undiscovered atomic bundle that gives S its value; i.e. $(T, v(T))$ is an atomic bid in the target XOR valuation, and of all atomic bids whose bundles are subsets of S, it has the greatest value. In the innermost loop, any element of $T \backslash S$ will be removed because the value remains unchanged: $(T, v(T))$ still remains the atomic bid that gives the resulting bundle its value. No element of T can be removed unless there is some other atomic bid $(T', v(T')$ such that $T' \subseteq S$ and $v(T') = v(T) = v(S)$. In this case the proof proceeds by replacing atomic bid $(T, v(T))$ with $(T', v(T'))$. As a result, the loop terminates with a bundle T that appears in an undiscovered atomic bid. We then add $(T, v(T))$ to the manifest valuation, and the induction hypothesis still holds. □

In the algorithm, the demand queries have the same effect as equivalence queries. Because the manifest valuation always maintains a subset of the actual set of atomic bids, it is always a lower bound on the target valuation. As a result, presenting the manifest together with the empty set in a demand query will always identify a counterexample unless all atomic bids have been identified.

The proof of correctness shows that the innermost loop identifies a new atomic bid in $O(m)$ time. This is repeated t times, where t is the number of atomic bids in the target valuation, so the worst-case runtime is polynomial in the relevant parameters.

This result should be contrasted with Blum et al.'s negative results ([15], Theorem 2) stating that monotone DNF (and hence XOR representations) cannot be learned efficiently when the demand queries are restricted to linear and anonymous prices over the goods.

Best-response sets

The learning algorithm for XOR is significant because it can be applied within established combinatorial auction designs such as Parkes' *i*Bundle [78], Ausubel and Milgrom's ascending-proxy auction [7], or de Vries et al.'s auction based on primal-dual methods [31]. At each round of each these auctions, bidders must provide their ϵ-best-response sets to the current prices, which are encoded with XOR. Here $\epsilon > 0$ is the price increment, meaning that if two bundles differ in price, the difference is a positive multiple of ϵ. The best-response sets should contain only bundles that appear in atomic bids of the valuations' XOR representations, and possibly the empty set. This is simple if bidders already have their valuations encoded with XOR. If not, then bidders need to translate their valuation information into the proper bid. An approach along the lines of Algorithm 2 allows one to do this using value and demand queries. It is given as Algorithm 3.

Lemma 6 *Algorithm 3 outputs the ϵ-best-response set with respect to prices p, such that each bundle in the set appears in an atomic bid of the target valuation's minimal XOR representation.*

Proof. The algorithm only adds bundles that are in the XOR representation of the target valuation to set b, by the same arguments as for Algorithm 2. Clearly, it also

Given: A value and demand query oracle for the valuation.

An XOR representation of prices p.

Output: The bundles in the ϵ-best-response set.

Let $b = \emptyset$.

Present $\emptyset$ and prices p as a demand query.

Find the utility (value minus price) $\bar{u}$ of the response via a value query.

repeat

Present $\emptyset$ together with prices p as a demand query.

Record the response S, or set $S := \emptyset$ if response was YES.

Find the utility u of S via a value query.

if $u \geq \bar{u} - \epsilon$ **then**

Set $T := S$.

foreach $j \in T$ **do**

Perform value queries on T and $T - j$.

if *the values are equal* **then**

Set $T := T - j$.

end

end

Add T to b.

Set $p := p \cup (T, p(T) + \epsilon)$.

end

until $u < \bar{u} - \epsilon$

Return the set b.

Algorithm 3: Learning algorithm for ϵ-best-response sets.

only adds to b bundles that are ϵ-best-responses, by the if-condition. So we must argue that all ϵ-best-responses are recovered by the algorithm.

Note that a bundle can only be added at most two times to set b. (Because b is a set, the second time a bundle is added, it is ignored.) Each time a bundle is added, its price increases by ϵ, so after two additions the bundle's utility with respect to the new prices must be 2ϵ less than $\bar{u}$. If the bundle is again encoutered beyond that point, the algorithm halts.

We say a bundle is "atomic" if it appears in an XOR bid of the target valuation, represented as XOR. We argue that a price increase does not rule out any atomic bundle except the one that sees the increase: if an atomic bundle T is an ϵ-best-response with respect to the initial p before the price increase on some atomic bundle $S \neq T$, then T is still an ϵ-best-response with respect to the initial p after the increase. To see this, first note that the price increase on bundle S does not change the price of T if $T \not\supset S$. If $T \supset S$, then it must be that $v(T) > v(S)$ since both are atomic bundles in the minimal XOR representation. As S was chosen over T by the demand query, the price of T must be strictly greater than the price of S. Since prices differ by multiples of ϵ, the price of T is then at least $p(S) + \epsilon$. This is the price of S after the increase, so the price of T cannot have changed.

This invariance means that every ϵ-best-response will be discovered at some point during the algorithm, which completes the proof. □

By using this subroutine within an auction such as *i*Bundle [78], it then becomes possible to compare the query complexity of the auction with the query complexity of other elicitiation schemes, such as the one we give in Chapter 5.

4.4.2 OR

Lahaie et al. [58] give a learning algorithm for OR representations that uses value and demand queries. At any given point in the algorithm, the atomic bids in the manifest valuation are a subset of the atomic bids in the target valuation's minimal OR representation.

Lemma 7 *Algorithm 4 outputs the minimal OR representation of the target valuation.*

Proof. Assume by induction that all atomic bids in the manifest valuation are also present in the target valuation's minimal OR representation. This is true initially, when the manifest valuation is set to the empty OR representation. The manifest valuation thus lower bounds the true valuation of any bundle, by OR semantics. If all atomic bids have been discovered, the empty set maximizes utility and will be returned upon a demand query, and the algorithm correctly halts. Otherwise, some other bundle S is returned. By OR semantics, the atomic bids that are selected to derive the value of S must contain at least one undiscovered atomic bid.

There are two cases. The first is where $(S, v(S))$ is an undiscovered atomic bid, and there are no undiscovered atomics $(T, v(T))$ such that $T \subseteq S$. By the construction of prices v', no bundle that is not a strict subset of S can be preferred to the empty set. Since all atomics $(T, v(T))$ such that $T \subseteq S$ have been discovered, no bundle that is a strict subset of S can be preferred to the empty set. Hence the empty set must be returned at this point, and $(S, v(S))$ is correctly added as a new atomic bid, and the induction hypothesis still holds.

Given: A value and demand query oracle for the valuation.

Output: The minimal OR representation of the target valuation.

Let v be the OR bid $\emptyset$.

(Throughout, v has OR semantics.)

repeat

  Present $\emptyset$ together with prices v as a demand query.

  Record the response S, or set $S := \emptyset$ if response was YES.

  Set $T := S$.

  repeat

    Let $v' := \{(\{j\}, +\infty) \mid j \notin T\} \cup (T, +\infty) \cup v$.

    Present $\emptyset$ together with prices v' as a demand query.

    if *the response is* YES **then**

      Record as w the result of a value query on T.

      Set $v := v \cup (T, w)$.

      Set T to $\emptyset$.

    end

    else

      Set T to the response.

    end

  until T *is* $\emptyset$

until S *is* $\emptyset$

Return v as an OR representation.

Algorithm 4: Learning algorithm for the OR language.

In the second case, there is an undiscovered atomic bid $(T, v(T))$ such that $T \subset S$. When prices v' are presented in the demand query, only strict subsets of S can have

positive value. Bundle T itself gives positive utility, so some counterexample $S' \subset S$ must be returned. Since the size of the counterexamples keeps strictly decreasing, at some point we must reach the first case. (Note that $\emptyset$ can never be a counterexample). □

The proof of correctness shows that the innermost loop identifies a new atomic bid in $O(m)$ time. This is repeated t times, where t is the number of atomic bids in the target valuation, so the worst-case runtime is polynomial in the relevant parameters.

Unfortunately, this algorithm is problematic because is requires us to present an OR bid as prices each time it performs a demand query. Recall that OR is unsuitable for pricing as of yet. Nevertheless, the algorithm is of theoretical interest because it is the only known learning algorithm for this language, and may prove useful in special cases where responding to OR demand queries is feasible (e.g. if we are willing to use a constraint-generation approach to evaluate prices in the OR language).

4.4.3 Polynomials

Schapire and Sellie [99] give a learning algorithm for sparse multivariate polynomials that uses value and equivalence queries. The equivalence queries made by the algorithm are all proper. Specifically, their algorithm learns the representation class of t-sparse multivariate polynomials over the real numbers, where the variables may take on values either 0 or 1. A t-sparse polynomial has at most t terms. A polynomial "over the real numbers" has coefficients drawn from the real numbers. So this corresponds exactly to the language described in Section 4.2.4.

The learning algorithm for polynomials makes at most $mt + 2$ equivalence queries

and at most $(mt + 1)(t^2 + 3t)/2$ value queries to a bidder, where t is the sparcity of the polynomial representing the bidder's valuation [99]. Because the algorithm makes equivalence queries, it is not ideal for use as a pre-processing step to encode a valuation before a one-shot auction. It can be applied in iterative schemes by switching the equivalence queries to demand queries, as described in Chapter 5.

In practice the algorithm can run slowly, because it requires a large number of value queries (cubic in the worst-case). However, there are ways to accelerate it. The algorithm repeatedly runs two important subroutine called `AddElement` and `EasyCounterexample` (see Schapire and Sellie [99] for the details of these subroutines). The first modifies the manifest valuation given a counterexample. The second searches for counterexamples to the manifest valuation by using only value queries (i.e. without performing an equivalence query). Each time a counterexample from an equivalence query is obtained, the manifest valuation is modified using `AddElement`, and then other counterexamples are obtained and incorporated by repeatedly running `EasyCounterexample` until no more counterexamples can be found via this method. The manifest valuation is then said to be 'stable.'

One way to speed up the computation is to only stabilize the manifest valuation after several equivalence queries are made, rather than after every single one. This means that certain equivalence queries may no longer be proper, but in practice this does not seem to be a concern—see the experimental evaluation in Chapter 5.

4.5 Other Languages

Nisan [74] lists several other languages such as XOR-of-OR, OR-of-XOR, and general OR/XOR formulae. These representations are natural next candidates for learning algorithms. However, the OR* language is as succinct as any of these [74], so the ideal would be to develop a learning algorithm for this language. Using the techniques of Chapter 5, this could then lead to powerful new methods for preference elicitation. A more immediate question though is whether there exists a learning algorithm for OR that does not require demand queries with OR prices.

The notion that a bidding language should allow for a MIP formulation of the value query problem is common in the literature. Boutilier [17] and Boutilier and Hoos [20] give bidding languages that use logical connectives to describe allowable combinations of atomic bids; the languages have natural formualtions as integer programs. The TBBL language by Cavallo et al. [22], suited to combinatorial exchanges [83], was also designed with an IP formulation in mind. These languages were all designed for bidding, but not for pricing.

Zinkevich et al. [111] were the first to apply the learning theory framework to the problem of encoding valuations. They consider the representation classes of *read-once formulae* and *Toolbox DNF*. Read-once formulae can represent certain substitutabilities, but not complementarities, whereas the opposite holds for Toolbox DNF. Since their work is also grounded in learning theory, they allow dependence on the size of the target valuation as we do (though read-once valuations can always be succinctly represented anyway). Their work only makes use of value queries, which are quite limited in power. Because we allow ourselves demand queries with expressive prices,

we are able to derive learning algorithms for the XOR and OR languages, which cover broader valuation classes than read-once formulae or Toolbox DNF.

Chapter 5

Preference Elicitation

We have seen how prices can be used as a market-clearing mechanism, and how learning algorithms can be used to efficiently encode agents' valuations. We are now in a position to leverage these tools for preference elicitation. For our purposes, "preference elicitation" refers to the process of extracting value information from the agents to the extent needed so that resources can be efficiently allocated, using a pre-defined, restricted set of query types. Note that this need not mean learning the valuations in full.

There are clear parallels between query learning and preference elicitation. In learning theory, the goal is to learn a function via various types of queries, such as value and equivalence queries. In preference elicitation, the goal is to elicit enough partial information about preferences to be able to compute an optimal allocation. Though the frameworks of learning and preference elicitation differ somewhat—the first involves a single agent, whereas the second occurs in a multi-agent setting—it is clear that these problems share similar structure, and it should not be surprising

that techniques from one field should be relevant to the other.

In this chapter, we show that any exact learning algorithm with value and equivalence queries can be converted into a preference elicitation algorithm with value and demand queries. The resulting elicitation algorithm guarantees elicitation in a polynomial number of value and demand queries. Here we mean polynomial in the number of goods, agents, and the sizes of the agents' valuation functions in a given encoding scheme. Preference elicitation schemes have not traditionally considered this last parameter. We argue that complexity guarantees for elicitation schemes should allow dependence on it. Introducing this parameter also allows us to guarantee polynomial worst-case communication complexity, which usually cannot be achieved in the number of goods and agents alone. The conversion procedure can be used to generate combinatorial auction protocols from learning algorithms for XOR, OR, and polynomial representations, using the learning algorithms from the previous chapter.

Learning theory is not concerned with incentives, because there is no larger goal of implementing an outcome. When moving to the setting of preference elicitation, incentives become relevant and we may want to extract enough information not only to implement an efficient outcome, but also to implement payments such as Vickrey payments to ensure agents are honest in responding to queries.

Section 5.1 introduces the preference elicitation framework, providing a formal definition of the notion of efficient elicitation. Section 5.2 then describes how learning algorithms can lead to elicitation algorithms, first by drawing a key analogy between equivalence and demand queries, and then by showing how learning algorithms can be embedded as subroutines into a preference elicitation protocol. The section

concludes with a way to implement the Vickrey outcome (allocation and payments) rather than the efficient allocation alone, as a way to induce agents to respond truthfully to queries. Section 5.3 discusses the communication requirements of preference elicitation protocols based on learning algorithms, in terms of the properties of those algorithms. Section 5.5 concludes the chapter with a discussion of the modularity of the whole approach.

5.1 Framework

In the intermediate rounds of a preference elicitation protocol, the auctioneer will have elicited information about the agents' valuation functions via various types of queries. She will thus have constructed a set of *manifest* valuations $\tilde{v}_1, \ldots, \tilde{v}_n$.[1] The values of these functions may correspond exactly to the true agent values, or they may be upper or lower bounds on the true values, depending on the types of queries made. They may also simply be default or random values if no information has been acquired about certain bundles. The goal in the preference elicitation problem is to construct a set of manifest valuations such that

$$\arg\max_{S \in \Gamma} \sum_{i \in N} \tilde{v}_i(S_i) \subseteq \arg\max_{S \in \Gamma} \sum_{i \in N} v_i(S_i).$$

That is, the manifest valuations provide enough information to compute an allocation that is optimal with respect to the true valuations. Note that we only require one such optimal allocation.

[1] This view of preference elicitation protocols is meant to parallel the learning setting. In many combinatorial auctions, for example, manifest valuations are not explicitly maintained but rather simply implied by the history of bids.

Two typical queries used in preference elicitation are *value* and *demand* queries, introduced in the previous chapter.[2] We make the following definition to parallel the query learning setting and to simplify the statements of later results:

Definition 2 *The representation classes $\mathcal{V}_1, \ldots, \mathcal{V}_n$ can be* ***efficiently elicited from value and demand queries*** *if there is a fixed polynomial $p(\cdot, \cdot)$ and an algorithm L with access to value and demand queries of the agents such that for any $v = (v_i)_{i \in N} \in \mathcal{V}_1 \times \ldots \times \mathcal{V}_n$, L outputs after $p(size(v), m)$ queries an allocation $S \in \arg\max_{S' \in \Gamma} \sum v_i(S'_i)$.*

There are some key differences here with the query learning definition. We have dropped the term "exactly" since the valuation functions need not be determined exactly in order to compute an optimal allocation. We also simply require a polynomial number of queries rather than polynomial time overall. Computing an optimal allocation of goods even when given the true valuations is NP-hard for a wide range of valuation classes. It is thus unreasonable to require polynomial time in the definition of an efficient preference elicitation algorithm. We are happy to focus on the communication complexity of elicitation because this problem is widely believed to be more significant in practice than that of winner determination [74].

Since the valuations need not be elicited exactly it is less clear whether the polynomial dependence on $size(v)$ is justified in this setting. We address this in the next section.

[2] Our definition of equivalence query differs slightly from the definition provided by Blum et al. [15] Their demand queries are restricted to linear prices over the goods. In contrast our demand queries allow for nonlinear prices. This is why the lower bound in their Theorem 2 does not contradict our results that follow.

5.2 From Learning to Preference Elicitation

5.2.1 Parallels between Equivalence and Demand Queries

We have described the query learning and preference elicitation settings in a manner that highlights their similarities. Both settings share value queries, and it turns out that equivalence and demand queries are analogs. Th key is to use CE prices. We saw in Chapter 3 that CE prices always exist for the CAP, although they may need to be non-anonymous and nonlinear. By Theorem 4, once we have identified an allocation together with supporting CE prices, the CAP is solved.

Recall from Chapter 3 that if π is a vector of core payoffs, then setting prices to

$$p_i(S) = \max\{0, \tilde{v}_i(S) - \pi_i\} \tag{5.1}$$

for all $i \in N$ and $S \subseteq M$ yields valid CE prices. These prices leave every agent indifferent across all bundles with positive price. Thus, both equivalence and demand queries can communicate the manifest valuations, the first explicitly and the second in the guise of discounted prices. Of course, CE prices do not necessarily have to be shifted valuations; for instance, linear CE prices may happen to exist. The following lemma shows how to obtain counterexamples to equivalence queries through demand queries.

Lemma 8 *Suppose an agent i replies with a preferred bundle S' when proposed a bundle S and supporting competitive equilibrium prices $p_i(S)$ (supporting with respect to the the agent's manifest valuation). Then either $\tilde{v}_i(S) \neq v_i(S)$ or $\tilde{v}_i(S') \neq v_i(S')$.*

Proof. We have the following inequalities:

$$\begin{aligned} & \tilde{v}_i(S) - p_i(S) \geq \tilde{v}_i(S') - p_i(S') \\ \Rightarrow \quad & \tilde{v}_i(S') - \tilde{v}_i(S) \leq p_i(S') - p_i(S) \end{aligned} \tag{5.2}$$

$$\begin{aligned} & v_i(S') - p_i(S') > v_i(S) - p_i(S) \\ \Rightarrow \quad & v_i(S') - v_i(S) > p_i(S') - p_i(S) \end{aligned} \tag{5.3}$$

Inequality (5.2) holds because the prices support the proposed allocation with respect to the manifest valuation. Inequality (5.3) holds because the agent in fact prefers S' to S given the prices, according to its response to the demand query. If it were the case that $\tilde{v}_i(S) = v_i(S)$ and $\tilde{v}_i(S') = v_i(S')$, these inequalities would contradict each other. Thus at least one of S and S' is a counterexample to the agent's manifest valuation. □

Finally, we justify dependence on $size(v)$ in elicitation problems. Intuitively, this is because we must learn valuations exactly when performing elicitation, in the worst-case. Nisan and Segal (Proposition 1, [75]) and Parkes (Theorem 1, [81]) show that supporting Lindahl prices must *necessarily* be revealed in the course of any preference elicitation protocol which terminates with an optimal allocation—see Theorem 11 for the exact statement and proof. Furthermore, Nisan and Segal (Lemma 1, [75]) state that in the worst-case agents' prices must coincide with their valuations (up to a constant), when the valuation class is rich enough to contain "dual valuations" (as will be the case with several interesting classes such as general or submodular valuations). Since revealing CE prices is a necessary condition for establishing an optimal allocation, and CE prices contain the same information as valuation functions

(in the worst-case), allowing for dependence on $size(v)$ in elicitation problems is entirely natural.

5.2.2 Learning as a Subroutine for Elicitation

The key to converting a learning algorithm to an elicitation algorithm is to simulate equivalence queries with demand and value queries until an optimal allocation is found. Given candidate CE prices, if all agents reply 'YES' to a demand query (i.e. accept the bundle proposed to them), then we have found an optimal allocation, analogous to the case where an agent replies 'YES' to an equivalence query when the target function has been exactly learned. Otherwise, we can obtain a counterexample to an equivalence query given an agent's response to a demand query.

Theorem 15 *The representation classes $\mathcal{V}_1, \ldots, \mathcal{V}_n$ can be efficiently elicited from value and demand queries if they can each be efficiently exactly learned from value and equivalence queries.*

Proof. The algorithm that uses learning algorithms as subroutines for preference elicitation is given as Algorithm 5. Consider step 4. If all agents accept their proposed bundles, then these bundles maximize the agents' utilities at the given prices. The allocation also maximizes the auctioneer's revenue at the given prices, because the prices support the allocation with respect to the manifest vlauations. Thus an optimal allocation has been found, by Theorem 4. Otherwise, at least one of S_i or S_i' is a counterexample to $\tilde{v}_i$, by Lemma 8. We identify a counterexample by performing value queries on both these bundles, and provide it to L_i as a response to its equivalence query.

Given: Efficient exact learning algorithms $(L_i)_{i \in N}$ for valuations classes $(\mathcal{V}_i)_{i \in N}$ respectively.

Output: An efficient allocation together with supporting CE prices.

repeat

Run algorithms L_i for all $i \in N$ in parallel until each requires a response to an equivalence query, or has halted with the agent's exact valuation.

Compute an optimal allocation $S = (S_i)_{i \in N}$ and supporting CE prices p with respect to the manifest valuations $(\tilde{v}_i)_{i \in N}$ determined so far.

To each $i \in N$, present (S_i, p) as a demand query.

if *all reply* YES **then**

Output (S, p) and halt.

end

else

foreach *agent i that did not reply* YES **do**

Let S_i' be the reply.

Perform value queries on S_i and S_i' to identify a counterexample.

Provide counterexample to L_i.

end

end

until *there is a signal to halt*

Algorithm 5: Converting learning algorithms to an elicitation algorithm.

This procedure will halt, since in the worst-case all agent valuations will be learned exactly. The procedure performs a polynomial number of queries, since $L_1, \ldots, L_n$ are all efficient learning algorithms. □

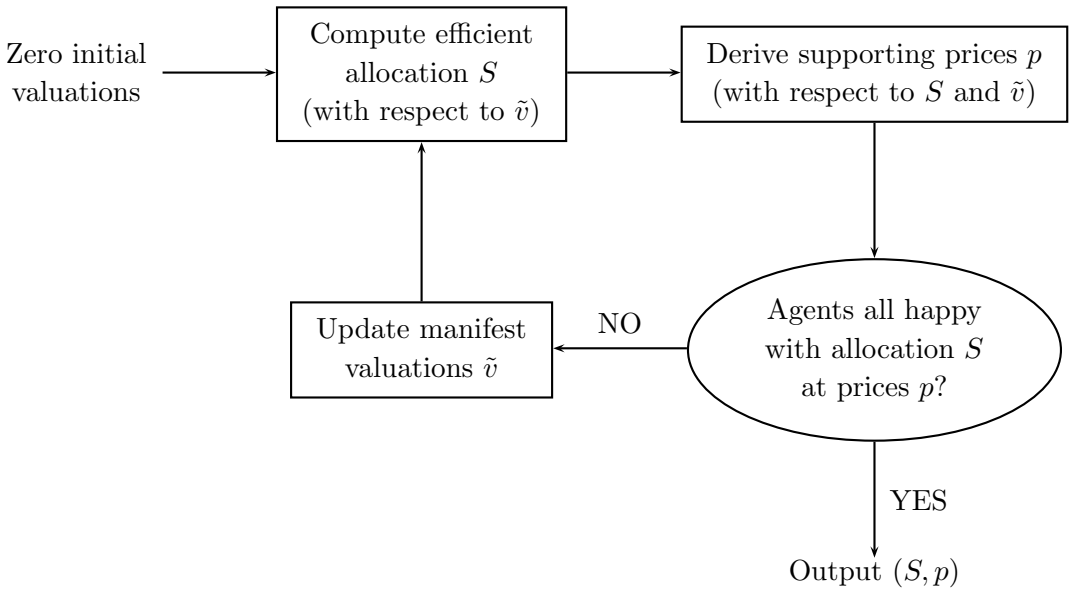


Figure 5.1: Flowchart of the elicitation and allocation process.

A flowchart of the resulting elicitation process is given in Figure 5.1. Note that the conversion procedure indeed results in a preference elicitation algorithm, not a learning algorithm. The resulting algorithm does not simply learn the valuations exactly, then compute an optimal allocation. Rather, it elicits partial information about the valuations through value queries, and periodically tests whether enough information has been gathered by proposing an allocation to the agents through demand queries. It is possible to obtain a competitive equilibrium for valuations $v = (v_i)_{i \in N}$ using an allocation and prices derived using manifest valuations $\tilde{v} = (\tilde{v}_i)_{i \in N}$, and finding an optimal allocation does not imply that the agents' valuations have been exactly learned. The use of demand queries to simulate equivalence queries enables this early halting once an optimal allocation has been found. We would not obtain this property with equivalence queries based on manifest valuations.

5.2.3 Incentives

As explained in Chapter 2, the VCG mechanism can be used to induce agents to respond truthfully to queries, while ensuring the auctioneer gets the highest possible revenue under the constraints that the allocation be efficient and that losing bidders pay nothing (recall Theorem 1). A naïve way to compute VCG payments is simply to run the protocol once with all agents, then once with each agent removed for a total of $n + 1$ runs. We would then determine the value of the efficient allocation in each run via value queries. This gives us sufficient information to compute VCG payments.[3]

In light of Theorem 10, we can instead modify the general elicitation framework to converge to UCE prices. In the first stage we run the standard elicitation protocol, until CE prices and an efficient allocation are determined. The second stage uses *universal demand queries*, a straightforward variant of demand queries.

Universal demand query. Bundles $(S_0, S_1, \ldots, S_n)$ together with prices p are input. The agent replies YES if all bundles presented maximize utility at prices p; otherwise some other utility-maximizing bundle is returned.

Whenever all the individual learning algorithms are stalled waiting to perform an equivalence query, we determine manifest allocations in the main economy (with agents N) and also in each of the marginal economies (with agents $N - j$, for all j). In addition, we determine (manifest) universal CE prices, p (for instance, we can take the manifest valuations themselves). We can then issue a universal demand query to

[3]Some care would be required to ensure consistency across runs, and also to "mix" queries up so that agents do not know the current run. Notice that an agent is indifferent in all runs except the one with all agents.

each agent i, by presenting the UCE prices together with the agent's allocation in the main economy and the marginal economies. If all agents reply YES, we have a UCE and can derive and implement VCG payments according to Theorem 10. Otherwise, by the same reasoning as in Lemma 8, we obtain counterexamples to push forward the individual learning algorithms.

As a result, we obtain an elicitation protocol that derives UCE prices and hence Vickrey payments in just one pass, rather than using $n + 1$ iterations of the original protocol. This approach is also interesting for experimentation purposes: the role of the queries performed in the second stage is precisely to compute Vickrey payments, so they represent the added query complexity of aligning incentives with the VCG mechanism.

5.3 Communication Complexity

We now turn to the issue of the communication complexity of elicitation. As mentioned, Nisan and Segal [75] show that for a variety of rich valuation spaces (such as general and submodular valuations), the worst-case communication burden of computing an efficient allocation is exponential in the number of goods, m. The communication burden is measured in terms of the number of bits transmitted between agents and auctioneer, in the case of discrete communication, or in the number of real numbers transmitted, in the case of continuous communication.

In such cases, it may still be possible to create preference elicitation protocols that make only a polynomial number of *queries* in m alone. However, these guarantees would be quite artificial because some of the queries themselves would have to contain

an exponential number of bits or real numbers. On the other hand, converting an efficient learning algorithm produces a preference elicitation algorithm whose queries have sizes polynomial in the parameters n, m, and $size(v)$, if the base learning algorithm's equivalence queries are all proper. Recall that an equivalence query is proper if $size(\tilde{f}) \leq size(f)$ at the time the query is made, where f is the target function and $\tilde{f}$ is the manifest hypothesis.

The size of any value query is $O(m)$: the message consists solely of the queried bundle. To communicate CE prices to agent j, it is sufficient to communicate the agent's manifest valuation function and possibly a price shift π_j; thus, if the base learning algorithm uses only proper equivalence queries, communicating the manifest valuation requires size $O(size(v_i))$. The surplus π_j to agent j cannot be any greater than $\max_{S \subseteq M} \tilde{v}_j(S)$, so communicating this value also requires size $O(size(v_i))$. We must also communicate to j its allocated bundle, so the total message size for a demand query is $O(size(v_i) + m)$. Clearly, an agent's response to a value or demand query is always $O(size(v_i) + m)$. The above discussion leads to the following result:

Theorem 16 *The representation classes $\mathcal{V}_1, \ldots, \mathcal{V}_n$ can be efficiently elicited from value and demand queries with polynomial communication complexity, in the parameters n, m, and $size(v)$, if they can each be efficiently exactly learned from membership and proper equivalence queries.*

Elicitation algorithms that depend on the $size(v)$ parameter sidestep Nisan and Segal's [75] negative results on the worst-case communication complexity of efficient allocation problems. They provide guarantees with respect to the sizes of the *instances* of valuation functions faced at any run of the algorithm. These algorithms

will fare well if the chosen representation class provides succinct representations for the simplest and most common of valuations, and thus the focus moves back to one of succinct yet expressive bidding languages.

5.4 Empirical Evaluation

An empirical evaluation of the framework provides several important insights. First, the theoretical results given above only provide worst-case bounds on query and communication complexity. In practice, the framework's performance may be better than these bounds suggest. Second, the theory gives no guidance on how to choose representations to use in the framework if we do not have detailed knowledge of the agents' valuations (the expected regime), so to do this we must appeal to intuition and empirical evidence. Third, the framework leaves open the possibility that valuations will not be fully learned before an efficient allocation is found. We would like to confirm that this indeed occurs, because otherwise the iterative approach would have little elicitation advantages over single-shot auctions.

To evaluate the empirical performance of the elicitation framework, I implemented it along with learning algorithms for the XOR and Polynomials languages (see Sections 4.4.1 and 4.4.3) for use by the proxies. The code is written in Java 1.5. I wrote the code with transparency and modularity in mind, and I did not perform any profiling or other such optimization beyond caching of value query responses, and the use of the server virtual machine rather than the client (`-server` option).[4] In the instantiated framework, winner determination is formulated as an integer program and

[4]The source code is available upon request.

solved using CPLEX 8.110. The experiments were run on a machine with a 1.86GHz Intel Core 2 Duo processor and 1GB of RAM, running Linux 2.6.9.

I generated agent valuations using benchmark distributions drawn from the literature on winner determination algorithms. This is appropriate because the distributions were designed to generate hard winner determination instances, or instances that are economically motivated, and the elicitation framework in essence provides a distributed algorithm for winner determination. The first three distributions generate an OR valuation for each individual agent by successively generating atomic bids until the desired size is reached. They differ only in how the size of an atomic bundle is determined. Once the size is fixed, the bundle's items are chosen uniformly at random without replacement. The value of the atomic bid is then drawn uniformly from a range that depends on the size of the atomic bundle.

Decay [2, 93, 32]. Repeatedly draw a number uniformly on $[0, 1]$ and increment the size k of the bundle until the random number exceeds a parameter $\alpha \in (0, 1)$. The value of the atomic bid is drawn uniformly from $[1, 1000k]$.

Exponential [2, 39]. The size of a bundle is k with probability $Ce^{-k/q}$, where C is a normalizing constant and q is a parameter that equals the expected size. The value of the atomic bid is drawn uniformly from $[500, 1500k]$.

Binomial [2, 39]. The size of a bundle is k with probability $\binom{m}{k}p^k(1-p)^{n-k}$. Here $p \in [0, 1]$ is a parameter and m is the number of items. This is equivalent to including each item in the bundle with probability p. The value of the atomic bid is drawn uniformly from $[500, 1500k]$.

An OR valuation is obtained by drawing atomic bids according to the chosen distribution until the representation is of the desired size. I also ensured that there were no redundant atomic bids, i.e. that the OR representation is minimal. To do this, we simplify the OR representation by removing each atomic bid in turn. We then check whether the value of its bundle remains unchanged; if so, the atomic bid is redundant and can be discarded. We then generate more atomic bids to again reach the desired size, and repreat the process until we obtain a minimal OR representation. This post-processing is more rigorous than simply removing "dominated" bids, which are bids (S_1, v_1) for which there exists another bid (S_2, v_2) such that $S_2 \subseteq S_1$ but $v_2 \geq v_1$ (this is standard post-processing [64]). Removing dominated bids may not result in a minimal representation with OR; on the other hand, it does guarantee a minimal representation with XOR.

The *Quadratic* distribution proposed by de Vries and Vohra [32] generates Polynomial representations of valuations. The representations consist of terms of size 1 and 2:

$$\sum_{j \in M} v_j y_j + \frac{1}{2} \sum_{j,k \in M' : k \neq j} v_{jk} y_j y_k,$$

where M' is a "synergy set" whose items are all complements of each other. The synergy set is obtained by drawing items uniformly without replacement until a set of size μ is reached, where μ is a parameter of the distribution. Thus the size of the polynomial representation increases as μ is increased, as there are more pairs of items that complement each other. Once M' is constructed, the coefficients v_j are drawn uniformly from $[0, 1]$ and $v_{jk} = v_j v_k$. In proposing this valuation model, deVries and Vohra [32] were guided by Ausubel et al.'s [6] description of the FCC spectrum

allocation problem.

Finally, I generated XOR representations using the *Arbitrary* distribution from Leyton-Brown et al.'s [64] CATS test suite. CATS provides several other distributions for generating XOR representations, and among these *Arbitrary* generates relatively hard instances [63]. As the name suggests, this distribution is meant to generate valuations that exhibit "arbitrary" complementarities and substitutabilities between bids. Unlike the previous distributions described, *Arbitrary* does not generate agent valuations individually; it can only generate several valuations at a time, given the total number of atomic bids desired across all representations. As a result, I cannot tune the number of agents or representation sizes with this distribution, only the total number of bids. The agent structure can be recovered because each atomic bid contains a "dummy item" specific to each agent. CATS only generates undominated bids, so the resulting XOR representations are minimal.

The instantiations of the framework all issued ϵ-demand queries, where ϵ was calibrated to achieve 99% efficiency on average for each vector of parameter settings. The data points in all plots are averages over 10 runs. I used 20 items for all experiments. A time limit of 1 hour was set for each run. If any of the 10 runs for a data point exceeded the time limit, computation was aborted and the data point was not plotted.

5.4.1 Scaling Performance

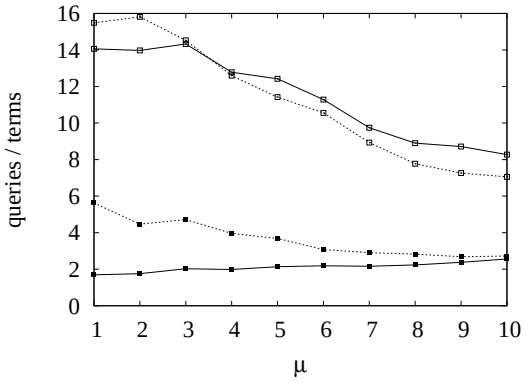
We first examine the query and runtime scaling properties of the framework with Polynomial representations at the proxy level, when the agents' underlying valuations

are generated with the *Quadratic* distribution and hence have a concise Polynomial representation.[5] I ran both "bidder-optimal" and "seller-optimal" versions. In the bidder-optimal version, valuations at each round were discounted by a bidder-optimal core payoff vector, and then quoted as prices. In the seller-optimal version valuations were quoted back directly as prices, corresponding to core payoffs where the seller extracts all the surplus.

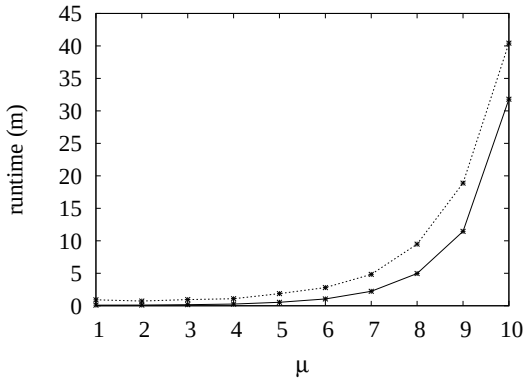
Figure 5.2(a) reveals that the query complexity of the elicitation process scales well with the size of the valuation's representation. The value queries per term decrease as the representations grow in size, whereas the demand queries remain essentially constant. This is much better than the cubic worst-case guarantees reported in Chapter 4 would suggest. Figure 5.2(b) shows that the runtime scales about linearly with the problem size (recall that as μ is scaled linearly, the instance sizes scale quadratically). This means winner-determination does not become substantially harder within the range $\mu = 1, \ldots, 10$.

The qualitative trends when scaling agents are shown in Figures 5.2(c)–(d), and are as expected. The number of queries, normalized by the number of terms, does not change as agents are scaled since the framework still performs about the same number of queries per agent. The runtime of the bidder-optimal version does not scale as well with the number of agents, because computing a bidder-optimal core payoff requires solving a winner-determination problem for the main and marginal economies.

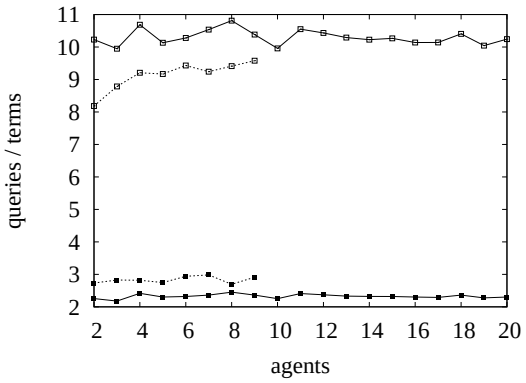
[5]It is important to understand that in the implementation, we use Polynomials only to model how the agents would respond to value and demand queries, and do not assume that the agents themselves are aware of these Polynomial representations. Proxies only obtain valuation information from the agents through an "oracle" interface of value and demand queries (this is an actual interface in the Java implementation). The same applies for agent valuations represented with OR or XOR as a result of different distributions.



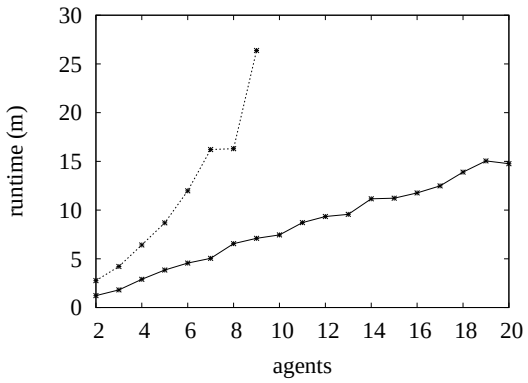
(a) Query complexity, quadratic distribution, scaling μ, with 3 agents.



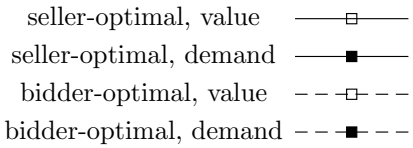
(b) Runtime, quadratic distribution, scaling μ, with 3 agents.



(c) Query complexity, quadratic distribution, scaling the number of agents, with $\mu = 7$.



(d) Runtime, quadratic distribution, scaling the number of agents, with $\mu = 7$.



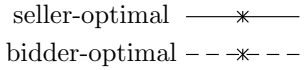
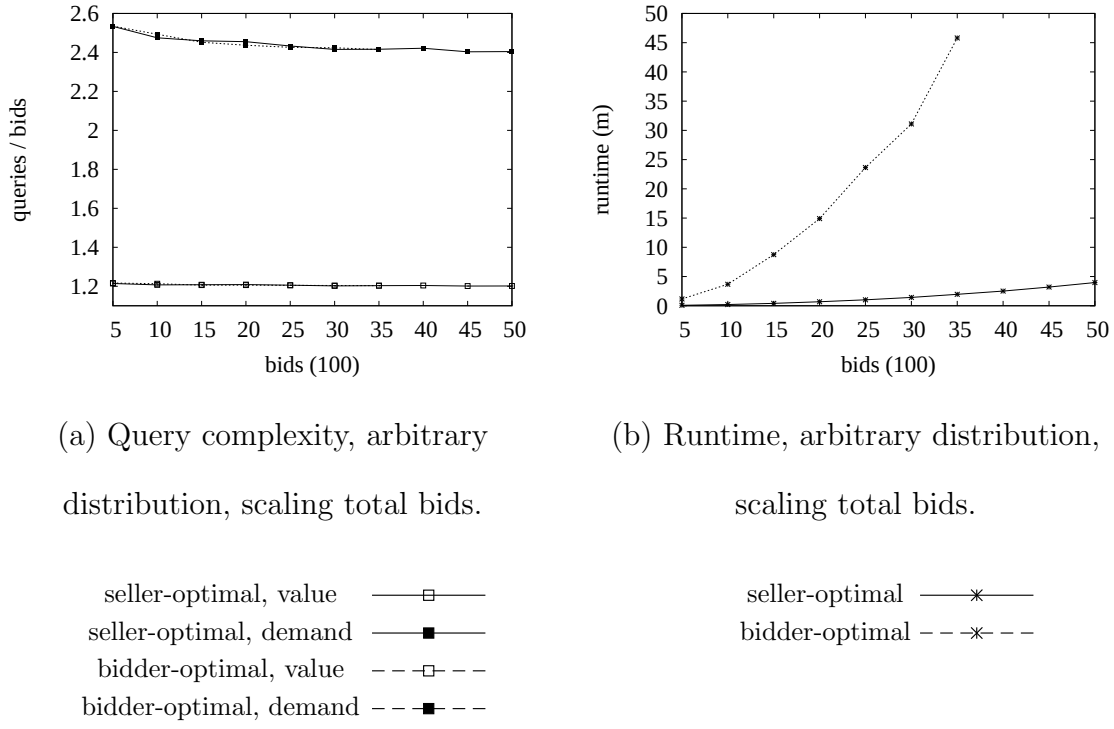


Figure 5.2: Scaling properties of the elicitation framework with Polynomial representations, with the quadratic distribution generating Polynomial representations of the bidders' valuations.



(a) Query complexity, arbitrary distribution, scaling total bids.

(b) Runtime, arbitrary distribution, scaling total bids.

Figure 5.3: Scaling properties of the elicitation framework with XOR representations, with the arbitrary distribution generating XOR representations of the bidders' valuations.

I tried the same kind of scaling experiment with XOR representations, when the agents had valuations generated from the *Arbitrary* distribution. This means the agents' valuations had concise XOR representations. The trends are largely similar to those for Polynomials, and are given in Figure 5.4.1. The number of queries per bid remains constant when scaling the number of bids, which means that queries scale linearly in the size of the instances. This agrees with the linear worst-case query complexity bounds for the XOR learning algorithm given in Section 4.4.1. The bidder-optimal version does not scale as well as the seller-optimal version in terms of runtime, because the number of agents grows large with the number of bids with the

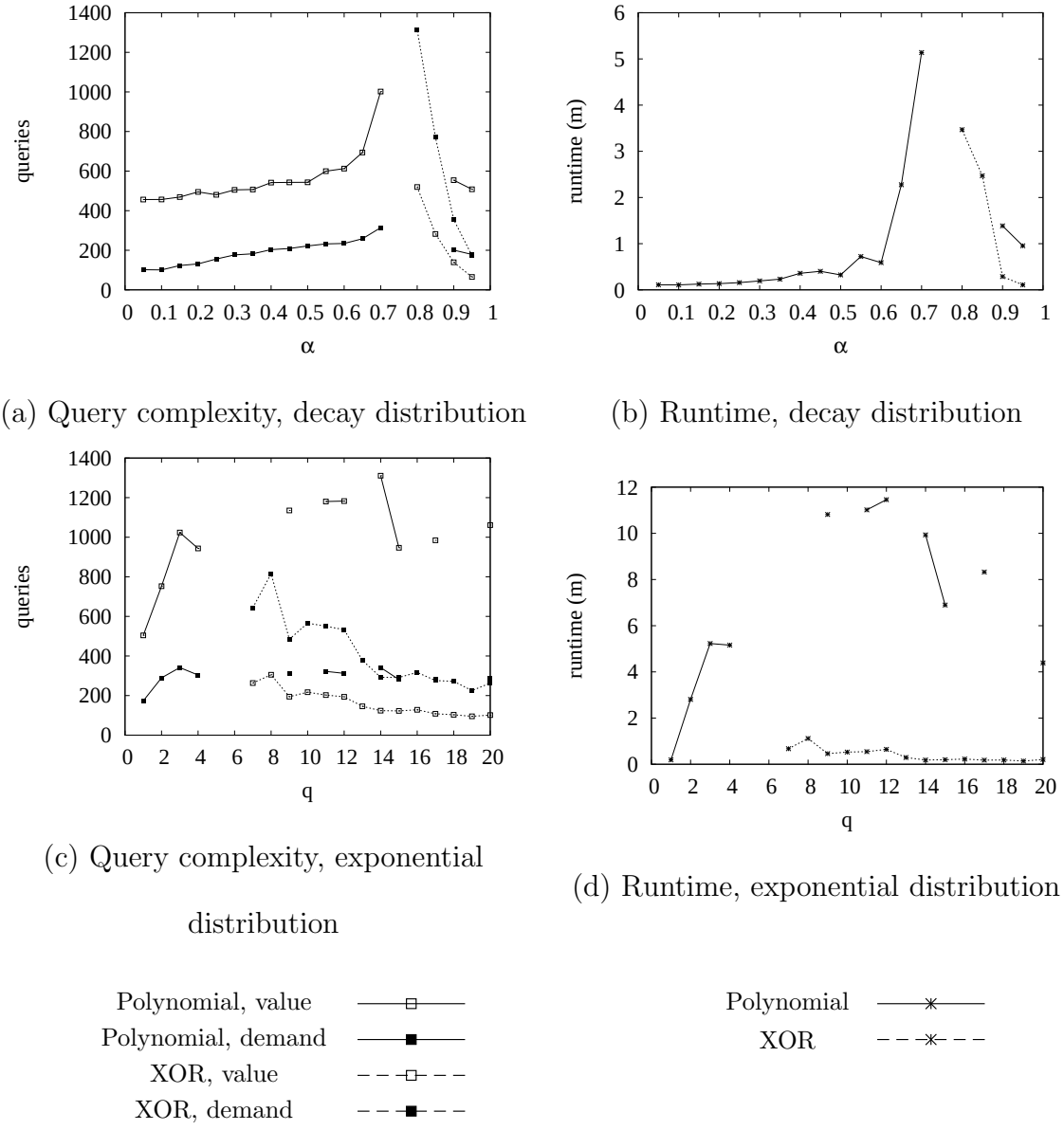
Arbitrary distribution.

I also implemented *i*Bundle(3), which uses nonlinear, non-anonymous prices, and gathered the same set of statistics. The results were extremely similar to the performance of the elicitation scheme with XOR representations, so they are not plotted for the sake of clarity. The close relationship is not surprising: the XOR elicitation scheme recovers the atomic bids of the agents' XOR representations in basically the same order as *i*Bundle.

5.4.2 Effect of Valuation Structure

The previous section examined the performance of the framework when the representations were ideally chosen: Polynomial representations when the agents themselves had valuations with concise Polynomial representations, and the analogous situation with XOR. A more realistic scenario is for the proxies' representations to be chosen based on just an intuitive understanding of the general structure of agent preferences. Here we examine the performance of the framework with Polynomial and XOR representations when the agents' valuations have concise OR representations.

If the bundles in an OR representation are relatively small, then there are many possibilities for combining them because many will be disjoint. We would therefore expect the resulting valuation to have more of an "additive" flavor, which could be succinctly represented with Polynomials. At the other extreme, if the bundles tend to be large, then they cannot be combined and we would expect the valuation to have a concise XOR representation. Though this intuition is quite crude, it agrees surprisingly well with the empirical findings. Figure 5.4(a)–(d) records the results.



(a) Query complexity, decay distribution

(b) Runtime, decay distribution

(c) Query complexity, exponential distribution

(d) Runtime, exponential distribution

Figure 5.4: Performance of the elicitation framework (seller-optimal version), with XOR and Polynomial representations, under different distributions for generating OR representations of the bidders' valuations. There are 3 agents and 15 atomic bids per agent.

With *Decay*, we see that Polynomials perform well when α is small—corresponding to small bundles—and even for larger settings such as 0.7, whereas XOR performs well

beyond 0.8. Polynomials complement XOR very well. I ran the elicitation scheme with XOR on $\alpha = 0.1$ with no time limit, and it had not terminated after 24 hours. One finding that I did not anticipate was that Polynomials also did well for settings of α beyond 0.9. In retrospect, it seems plausible that OR representations with such large bundles would correspond to valuations that also have succinct Polynomial representations: the atomics in an OR representation could be terms in the Polynomial, and then there are just a few possible higher-order terms that might be needed to complete the representation.

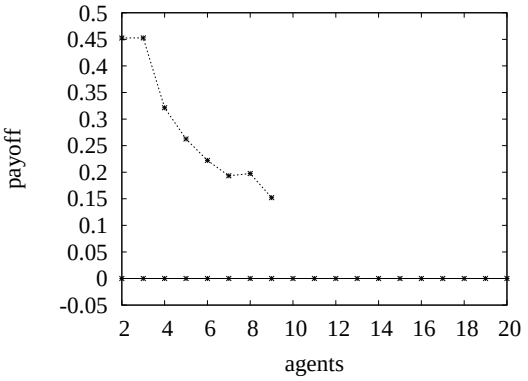
Both did poorly at $\alpha = 0.75$. If we could use OR representations, the resulting elicitation scheme would clearly perform well for all α, since the underlying OR representations are small. Recall that we cannot as of now use the OR learning algorithm within the framework because OR is not a suitable pricing language.

With the *Exponential* distribution, the same intuition applies, except that the elicitation scheme with XOR performs better than the version with Polynomials over a wider range of paramters. The results for the *Binomial* distribution were similar to those for the *Exponential* distribution and are omitted for brevity.

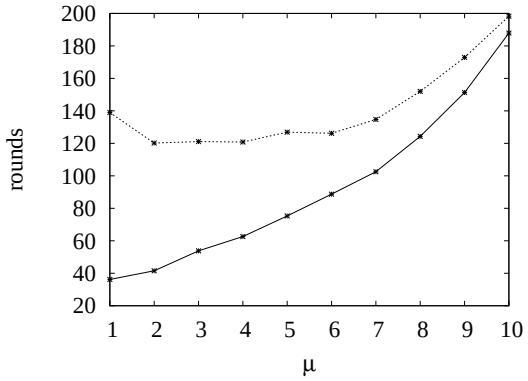
5.4.3 Surplus Distribution

The scaling results given above may be more favorable for the seller-optimal versions of the elicitation schemes, but note that they are only valid under the assumption of myopic best-response bidding. In the seller-optimal versions, the seller extracts all the surplus if agents bid myopically, so we would not expect this to happen. Further elicitation would have to be done to converge to UCE prices, from which we could

then derive VCG payments, and bring myopic best-responses into an equilibrium (see Section 5.2.3). The bidder-optimal versions do not bring myopic best-response bidding into an equilibrium either. On the other hand, we discussed in Chapter 3 how implementing bidder-optimal core payoffs could minimize the agents' incentives to misreport their preferences, so a bidder-optimal version may have adequate incentive properties in some settings even without extension to UCE prices, whereas a seller-optimal version would never have good incentive properties.



(a) Total agent payoff (as a percent of total surplus), scaling agents, with $\mu = 7$.



(b) Total rounds, scaling μ, 3 agents.

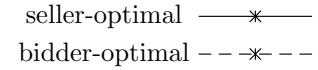
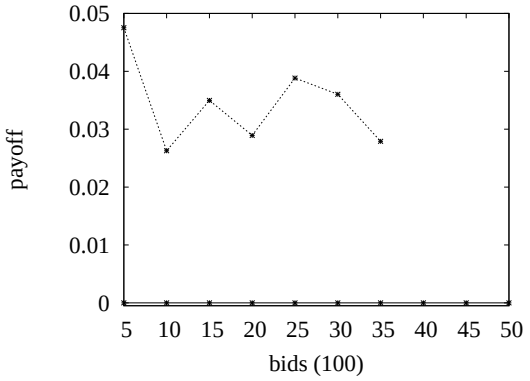


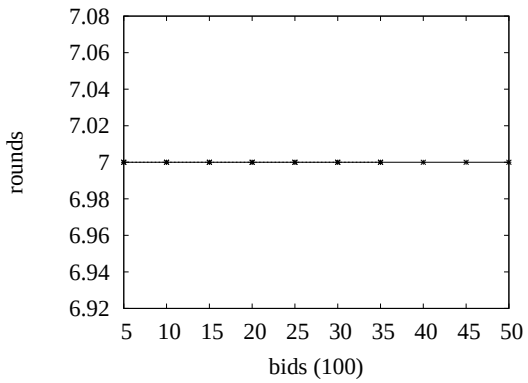
Figure 5.5: Bidder payoffs and total rounds in the elicitation framework with Polynomial representations, with the quadratic distribution generating Polynomial representations of the bidders' valuations.

To illustrate the advantage of bidder-optimal payoffs, consider Figure 5.5. As expected, the seller extracts all the surplus in the seller-optimal version of the elicitation scheme with Polynomial valuations. With the bidder-optimal version, the

bidders may be able to obtain around 30% of the total surplus when their number is small, i.e. 2–5. The seller-optimal version needs much fewer rounds, however. This is because more information is elicited at each round: in the bidder-optimal version, there is a greater chance for agents to be satisfied with their allocations at any given round, in which case no further information is elicited, even though it may be required at a later round anyway.



(a) Total agent payoff (as a percent of total surplus), scaling bids.



(b) Total rounds, scaling bids.

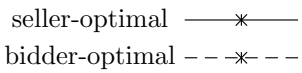


Figure 5.6: Bidder payoffs and total rounds in the elicitation framework with XOR representations, with the arbitrary distribution generating XOR representations of the bidders' valuations.

Figure 5.6 gives the same statistics for the elicitation scheme that uses XOR representations. In this case, though, the possible bidder surplus is very small. The reason is that with thousands of bids, the *Arbitrary* distribution implicitly generates a very large number of agents, so the degree of competition precludes much surplus

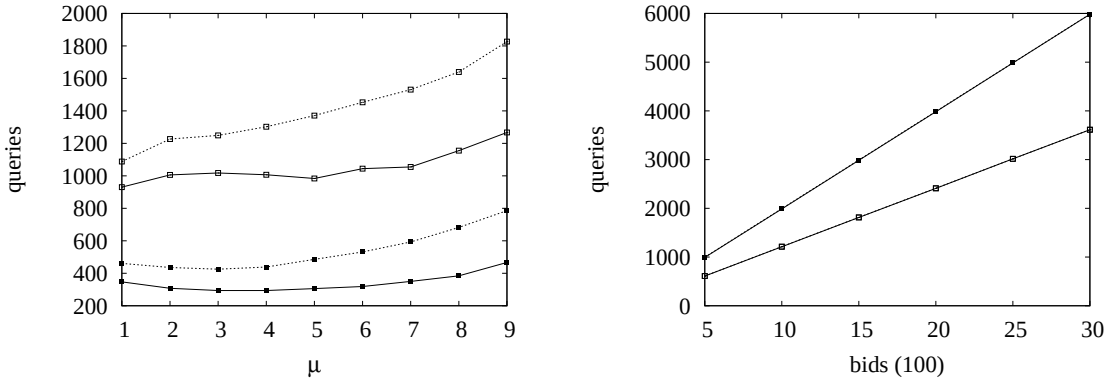
going to the bidders. The number of rounds, on the other hand, stays constant. This is the case because even though the number of bids and agents scales up, the typical size of an agents' XOR valuation under the *Arbitrary* distribution remains constant, and this is what affects the number of rounds.

5.4.4 Extent of Learning

The elicitation framework allows for the possibility that valuations may not be learned entirely before an efficient allocation (together with supporting prices) is found. If valuations were learned entirely, there would be no real improvement over the single-shot case: valuations could just as well be learned beforehand as a pre-processing step for a single-shot auction. To evaluate whether valuations are learned entirely, I ran the elicitation framework using Polynomial and XOR representations, with valuations generated with the *Quadratic* and *Arbitrary* distributions, respectively. Once each run was finished, the proxies then learned their agents' valuations, starting from the manifest valuations they ended up with after the elicitation had halted.[6]

Figure 5.7 presents the results for bidder-optimal instantiations of the framework. The fact that valuations need not be learned entirely is clear from Figure 5.7(a): a nontrivial amount of value and demand queries are needed to learn the valuations beyond the manifests obtained through elicitation. The number of additional value

[6]Alternatively, we could run the elicitation and learning algorithms separately. This turns out to be problematic for comparison purposes, because the learning may in fact require less queries than the elicitation. The reason for this is that the sequences of value and demand queries may be very different in the two approaches. The responses in the learning scheme may happen to be less adversarial than in the elicitation scheme, resulting in fewer queries. The opposite case may also occur.



(a) Polynomial representations, quadratic distribution, 3 agents, scaling μ.

(b) XOR representations, arbitrary distribution, 3 agents, scaling bids.

Figure 5.7: Additional learning queries required beyond the queries performed in the elicitation framework.

queries required is particularly informative. At $\mu = 1$, learning requires 17% more value queries beyond elicitation, and the gap increases so that at $\mu = 9$, 44% more value queries are needed. This kind of improvement is particularly important if responding to value queries is difficult for agents, e.g. if it involves solving an NP-hard optimization problem.

In Figure 5.7(b), on the other hand, we see no improvement at all (only two lines seem to appear because the value queries for elicitation and learning coincide, and similarly for the demand queries). This does not seem to be a defect of elicitation with XOR, but rather an artifact of the *Arbitrary* distribution. As we scale the number of bids, the XOR representations generated do not grow in size, but rather more agents

are created. In fact, the distribution tends to create small XOR representations, including a large number of single-minded valuations. If valuations are small there is a good chance they will be learned almost entirely in the elicitation phase, especially since a large number of agents implies that there will be a large number of losing bidders.

5.5 Discussion

We have seen that exact learning algorithms with value and equivalence queries can be used as a basis for preference elicitation algorithms with value and demand queries. At the heart of this result is the fact that demand queries may be viewed as modified equivalence queries, specialized to the problem of preference elicitation. The result allows us to apply the wealth of available learning algorithms to the problem of preference elicitation.

A learning approach to elicitation also motivates a different approach to designing elicitation algorithms that decomposes neatly across agent types. If the designer knowns beforehand what types of preferences each agent is likely to exhibit (mostly additive, many substitutes, etc...), she can design learning algorithms tailored to each agents' valuations and integrate them into an elicitation scheme. The resulting elicitation algorithm makes a polynomial number of queries. If the base learning algorithms' equivalence queries are all proper, the elicitation algorithm also has polynomial communication complexity.

I do not claim that agent valuations can be *learned* with value and demand queries; equivalence queries can only be simulated up to the point where an optimal allocation

has been computed. This is the preference elicitation problem. Theorem 15 implies that elicitation with value and demand queries is no harder than learning with value and equivalence queries, but does not provide any asymptotic improvements over the learning algorithms' complexity. It would be interesting to find examples of valuation classes for which elicitation is *easier* than learning. Blum et al. [15] provide such an example when considering value queries only (their Theorem 4).

Chapter 6

Iterative Auctions

The standard preference elicitation scheme for the combinatorial allocation problem is the iterative auction. Iterative auctions operate by issuing demand queries, or variants of demand queries where entire best-response sets are reported. In this chapter, we design an iterative auction that begins with linear prices and introduces nonlinearities into the price formulation as required in order to ensure the existence of a competitive equilibrium. The auction quotes pseudo-additive prices at each round.[1] Since the pseudo-additive language can represent some prices succinctly when the XOR language cannot (the most immediate example being linear prices), our auction design complements existing designs such as Parkes' *i*Bundle [78], Ausubel and Milgrom's ascending-proxy auction [7], and deVries et al.'s auction based on primal-dual methods [31].

The CAP can be formulated as a mathematical program. The advantage of this is that certain algorithms for solving the program can then be given auction inter-

[1]See Section 4.2.5 for a description of the pseudo-additive language.

pretations [11, 31, 32]. Bikhchandani et al. [11] outline and illustrate the approach. First, the CAP is formulated as a linear program that has an integer optimal solution. The dual variables of the formulation have natural interpretations as prices; therefore, decentralized dual methods for the LP that reach an integer solution can be given auction interpretations. de Vries et al. [31] give an auction interpretation to the primal-dual algorithm on an LP formulation for the CAP due to Bikhchandani and Ostroy [13], and note that *i*Bundle [78] (or equivalently, the ascending-proxy auction [7]) is a subgradient algorithm for this same formulation. In fact, the design of *i*Bundle was inspired by a primal-dual algorithm of Bertsekas [8] for the assignment problem, which he called the "auction algorithm."

We will use the LP approach to design our auction. In the approach given by Bikhchandani et al. [11], an integral LP formulation is given a priori. The problem is that without detailed knowledge about the underlying valuations (e.g. knowledge about substitutes or complements properties), there may be no choice but to use a very general formulation, which implies a high-dimensional price space. For instance, suppose the agents all have additive valuations. Then a simultaneous ascending auction is efficient and finds a CE with linear prices. But if the auctioneer cannot be sure that the valuations are all additive, he may resort to *i*Bundle [78] instead to ensure efficiency. As a result, the auction could require a very large amount of communication, because the XOR best-response sets at each round could be huge: the XOR representation of an additive valuation is exponential in m, and in the worst-case an entire valuation may be discovered.[2]

One solution to achieve a compact representations for prices may be to use the

[2]This follows from Theorem 11 due to Nisan and Segal [76].

method of the previous chapter. However, this method does not work if a proxy's manifest is represented in a bidding language but not a pricing language (e.g. in the OR language). Also, the method may be problematic due to privacy concerns: if the prices are made public (e.g. for the purpose of proving that the final allocation is indeed efficient), then everyone is aware of a succinct representation of each bidder's partial value information.

What we would like is a way to compute a succinct representation of CE prices that does not necessarily bear relation to the proxies' representations. In the example mentioned above, it would be ideal to begin with linear prices, and have the option to introduce nonlinearities as needed to ensure existence of a CE. Our idea in this chapter is to use the simplex method with column generation—which can be given an auction interpretation—to solve an LP formulation for the CAP, and then to use cutting planes to strengthen the formulation if the solution is not integer. This is equivalent to gradually introducing nonlinearities in the price space, where prices are represented in the pseudo-additive language. de Vries and Vohra [32] introduced both the ideas of using column generation and cutting-plane techniques in combinatorial auction design. Here we apply these ideas to obtain a specific auction design. In one variant, the design works as a stand-alone auction, and in another it can be used as a subroutine to compute CE prices at each round in the preference elicitation method of the previous chapter.

Section 6.1 introduces three LP formulations that form the basis of auctions that use order 1, 2, and 3 prices. Section 6.2 then uses the notion of a pattern (see Section 4.2.5) to define intermediate formulations between orders 1 and 2. Section 6.3

describes how to solve the LP formulations using the simplex method with delayed column generation, and how this can be interpreted as an auction. Section 6.4 explains how cutting planes can be introduced to strengthen a pattern formulation, and how this changes the price representation. Section 6.5 concludes with a discussion of other auctions and how they relate to the concepts in this chapter.

6.1 Three Formulations

Following the approach outlined by Bikhchandani et al. [11], the first step is to formulate the CAP as a linear program.[3] Figure 6.1 gives three separate LP formulations for the CAP together with their duals. These formulations were first given by Bikhchandani and Ostroy [13]. In all programs, variable $y_i(S)$ is used to indicate whether agent i receives bundle S. In the second program we also have a variable $z(\omega)$ to indicate whether the items are partitioned according to $\omega \in \Omega$, and similarly in the third formulation there is a variable $z(\gamma)$ to indicate that allocation $\gamma \in \Gamma$ is chosen.

The first constraint in all primal programs matches supply with demand, ensuring that the agents do not obtain any more than what is supplied. The next constraints are demand-side constraints that ensure each agent only gets at most one bundle. Finally, the last constraint in the second and third formulations ensures that only one partition or allocation is chosen.

In the duals, the variables corresponding to the first set of constraints in the

[3]This chapter makes use of several standard facts from linear programming theory. See any introductory linear programming textbook for a reference to these results [9, 24, 66].

$$\max_{y_i(S),z(\omega)} \quad \sum_{S\subseteq M}\sum_{i\in N} v_i(S)y_i(S)$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_{S\ni j}\sum_{i\in N} y_i(S) \le 1 && (j \in M)\\
& \sum_{S\subseteq M} y_i(S) \le 1 && (i \in N)\\
& y_i(S) \ge 0 && (i \in N, S \subseteq M)
\end{aligned}$$

(A) order 1 primal

$$\min_{\pi_i,p_j} \quad \sum_{i\in N}\pi_i + \sum_{j\in M} p_j$$

$$\begin{aligned}
\text{s.t.} \quad & \pi_i \ge v_i(S) - \sum_{j\in S} p_j && (i \in N, S \subseteq M)\\
& \pi_i \ge 0 && (i \in N)
\end{aligned}$$

(a) order 1 dual

$$\max_{y_i(S),z(\omega)} \quad \sum_{S\subseteq M}\sum_{i\in N} v_i(S)y_i(S)$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_{i\in N} y_i(S) \le \sum_{\omega\ni S} z(\omega) && (S \subseteq M)\\
& \sum_{S\subseteq M} y_i(S) \le 1 && (i \in N)\\
& \sum_{\omega\in\Omega} z(\omega) \le 1\\
& y_i(S) \ge 0 && (i \in N, S \subseteq M)\\
& z(\omega) \ge 0 && (\omega \in \Omega)
\end{aligned}$$

(B) order 2 primal

$$\min_{\pi_i,\pi^s,p(S)} \quad \sum_{i\in N}\pi_i + \pi^s$$

$$\begin{aligned}
\text{s.t.} \quad & \pi_i \ge v_i(S) - p(S) && (i \in N, S \subseteq M)\\
& \pi^s \ge \sum_{S\in\omega} p(S) && (\omega \in \Omega)\\
& \pi_i \ge 0 && (i \in N)\\
& \pi^s \ge 0
\end{aligned}$$

(b) order 2 dual

$$\max_{y_i(S),z(\gamma)} \quad \sum_{S\subseteq M}\sum_{i\in N} v_i(S)y_i(S)$$

$$\begin{aligned}
\text{s.t.} \quad & y_i(S) \le \sum_{\gamma:\gamma_i=S} z(\gamma) && (i \in N, S \subseteq M)\\
& \sum_{S\subseteq M} y_i(S) \le 1 && (i \in N)\\
& \sum_{\gamma\in\Gamma} z(\gamma) \le 1\\
& y_i(S) \ge 0 && (i \in N, S \subseteq M)\\
& z(\gamma) \ge 0 && (\gamma \in \Gamma)
\end{aligned}$$

(C) order 3 primal

$$\min_{\pi_i,\pi^s,p_i(S)} \quad \sum_{i\in N}\pi_i + \pi^s$$

$$\begin{aligned}
\text{s.t.} \quad & \pi_i \ge v_i(S) - p_i(S) && (i \in N, S \subseteq M)\\
& \pi^s \ge \sum_{i\in N} p_i(\gamma_i) && (\gamma \in \Gamma)\\
& \pi_i \ge 0 && (i \in N)\\
& \pi^s \ge 0
\end{aligned}$$

(c) order 3 dual

Figure 6.1: Three orders of linear programming formulations for the CAP and their duals.

primals can be interpreted as prices. We call the formulations "order 1, 2, and 3" formulations because we see from Figure 6.1 that the order k primal leads to order k prices in the dual, for $k \in \{1, 2, 3\}$. The objective in the dual is to set prices so as to minimize the total surplus to the bidders and auctioneer, where the surplus to a bidder is the maximum utility it can obtain over all bundles, and the surplus to the auctioneer is the maximum revenue it can obtain over all allocations. In the order 1 formulation, all allocations that give away all the items maximize revenue because we have order 1 prices, whereas in the order 2 formulation, any two allocations that are permutations of each other yield the same revenue because we have order 2 prices.

Any feasible allocation γ has a corresponding feasible integer solution to the LP in each formulation. We set $y_i(S) = 1$ if i obtains S in the allocation, and 0 otherwise. In the order 2 formulation, we set $z(\omega) = 1$, where ω is the partition induced by γ, and in the order 3 formulation we set $z(\gamma) = 1$. All other z variables are set to 0. Conversely, every feasible integer solution can be mapped to the allocation that gives S to i if and only if $y_i(S) = 1$. The order 1 and 2 formulations do not necessarily have an optimal integer solution, whereas the third does. The question of whether there exists an optimal solution for each formulation is closely related to the question of whether there exists CE prices of the corresponding order.

Proposition 5 [12, 13] *The order k primal formulation has an integer optimal solution if and only if order k CE prices exist for the given profile of valuations, for $k \in \{1, 2, 3\}$.*

Proof. We prove the proposition for the order 2 formulation. The other cases are entirely analogous. Let (x, z) and (π, y) be feasible solutions to the primal and dual

problem, respectively, and assume that (x, z) is integer. The solutions are optimal for the two respective problems if and only if they satisfy the primal complementray slackness conditions,

$$p(S) > 0 \Rightarrow \sum_{i \in N} y_i(S) = \sum_{\omega \ni S} z(w) \qquad (S \subseteq M) \tag{6.1}$$

$$\pi_i > 0 \Rightarrow \sum_{S \subseteq M} y_i(S) = 1 \qquad (i \in N) \tag{6.2}$$

$$\pi^s > 0 \Rightarrow \sum_{\omega \in \Omega} z(\omega) = 1 \tag{6.3}$$

as well as the dual complementray slackness conditions,

$$y_i(S) > 0 \Rightarrow \pi_i = v_i(S) - p(S) \qquad (i \in N, S \subseteq M) \tag{6.4}$$

$$z(\omega) > 0 \Rightarrow \pi^s = \sum_{S \in \omega} p(S) \qquad (\omega \in \Omega) \tag{6.5}$$

So assume (x, z) is an integer optimal primal solution and (π, y) an optimal dual solution. Consider the allocation where agent i gets S if $y_i(S) = 1$, and gets $\emptyset$ if y_i is the zero vector. We can assume that the partition ω for which $z(\omega) = 1$ corresponds to this allocation, because the solution is feasible in this case and still optimal (note that only y variables appear in the primal objective). If all y_i are zero, then we can also take z to be the zero vector. By condition (6.4), agent i's bundle maximizes the agent's utility at prices p, for all $i \in N$. By condition (6.5), the chosen partition maximizes the auctioneer's revenue at prices p. Hence prices p support the allocation just constructed.

Conversely, assume there exist supporting CE prices p for an allocation. We set $y_i(S) = 1$ if agent i receives S in the allocation, and the remaning components of y_i to 0. We set $z(\omega) = 1$ for the partition ω that corresponds to our allocation, and the remaining components of z to zero. Finally, we set $\pi_i = \max_{S \subseteq M} [v_i(S) - p(S)]$

and $\pi^s = \max_{\omega \in \Omega} \sum_{S \in \omega} p(S)$. Clearly, these primal and dual solutions are feasible. By construction of the primal solution, condition (6.1) holds. Condition (6.2) holds because $\pi_i = 0$ if i receives $\emptyset$, from the fact that p supports the allocation. Similarly, condition (6.3) holds because $\pi^s = 0$ if the auctioneer allocates nothing. The dual complementary slackness conditions hold by the fact that p are supporting prices. Hence the primal and dual are optimal, and the primal has an integer optimal solution. □

Referring back to the results in Table 3.1, we see then that the order 1 formulation has an integer optimal solution if the bidders have substitutes valuations, the order 2 formulation has an integer optimal solution if the bidders have superadditive valuations, and the order 3 formulation has an integer optimal solution if the bidders have general valuations (i.e. always, in our model).

Note that there is a large gap, however, between the order 1 and 2 formulations. The first has a polynomial number of constraints, whereas the second has a number of constraints that is exponential in m. To obtain prices of a more manageable size, we need a formulation that is in some sense "intermediate" between the order 1 and 2 formulations.

6.2 Pattern Formulation

Patterns can be used to describe CAP formulations that are intermediate between orders 1 and 2. With a slight abuse of notation, let $\mathcal{B}(S)$ denote the decomposition of bundle S according to pattern $\mathcal{B}$, following Algorithm 1. For $T \in \mathcal{B}$, let $\mathcal{B}^{-1}(T)$ be the set of bundles S such that $T \in \mathcal{B}(S)$, i.e. those bundles that have T in their

decomposition. The linear programming formulation is as follows.

$$\begin{aligned}
\max_{y_i(S),z(\omega)} \quad & \sum_{S\subseteq M}\sum_{i\in N} v_i(S)y_i(S) & & \\
\text{s.t.} \quad & \sum_{S\in\mathcal{B}^{-1}(T)}\sum_{i\in N} y_i(S) \leq \sum_{\omega:\omega\cap\mathcal{B}^{-1}(T)\neq\emptyset} z(\omega) & (T\in\mathcal{B}) & (6.6)\\
& \sum_{S\subseteq M} y_i(S) \leq 1 & (i\in N) & (6.7)\\
& \sum_{\omega\in\Omega} z(\omega) \leq 1 & & (6.8)\\
& y_i(S) \geq 0 & (i\in N, S\subseteq M) & \\
& z(\omega) \geq 0 & (\omega\in\Omega) &
\end{aligned}$$

For any pattern $\mathcal{B}$, this formulation has an integer solution corresponding to each feasible allocation $S = (S_i)_{i\in N}$. We set $y_i(S_i) = 1$ and all other components of y_i to 0, for all $i \in N$. We set $z(\omega) = 1$ for $\omega = \{S_i\}_{i\in N}$ and all other components of z to 0. Clearly this solution satisfies constraints (6.7) and (6.8). Let $T \in \mathcal{B}$ and consider the constraint corresponding to T in (6.6). This constraint cannot have both $y_i(S_i)$ and $y_j(S_j)$ on the left-hand side, for $i \neq j$. If it did, then T would be in the decompositions of both S_i and S_j, and hence we would have $S_i \cap S_j \supseteq T \neq \emptyset$, contradicting the fact that S was a feasible allocation. If the left-hand side has just $y_i(S_i)$ for some $i \in N$, the right-hand side has $z(\omega)$, because $S_i \in \omega$ and $S_i \in \mathcal{B}^{-1}(T)$. Hence the constructed integer solution is feasible.

The pattern formulation specializes to the order 1 and 2 formulations mentioned previously. Taking the pattern that consists solely of the singletons $\{j\}$ for $j \in M$ yields the order 1 formulation, because each bundle is decomposed into its constituent items (in this case the order of the singletons in the pattern is irrelevant). Taking the pattern that consists of all bundles $S \subseteq M$ yields the order 2 formulation, because each

bundle decomposes to itself (again the order of the bundles in the pattern is irrelevant in this case). When specializing to the order 1 formulation, the $z(\omega)$ variables always appear all together, so constraint (6.8) can be dropped and the right-hand-side of constraints (6.6) simplifies to 1.

The dual of the CAP formulation for a generic pattern $\mathcal{B}$ is as follows.

$$\begin{aligned}
\min_{\pi_i, \pi^s, p(T)} \quad & \sum_{i \in N} \pi_i + \pi^s & & \\
\text{s.t.} \quad & \pi_i \geq v_i(S) - \sum_{T \in \mathcal{B}(S)} p(T) & (i \in N, S \subseteq M) & \quad (6.9) \\
& \pi^s \geq \sum_{S \in \omega} \sum_{T \in \mathcal{B}(S)} p(T) & (\omega \in \Omega) & \quad (6.10) \\
& \pi_i \geq 0 & (i \in N) & \\
& \pi^s \geq 0 & &
\end{aligned}$$

Note that the patterns that lead to the order 1 and 2 formulations lead to order 1 and 2 prices in the dual, as expected.

By incrementally adding bundles to a pattern, starting with just the singletons $\{j\}$ for $j \in M$ and ending with all bundles, we obtain a series of formulations intermediate to the order 1 and 2 formulations. This leads to a dynamic way to adapt the price representation in an auction so as to ensure that a competitive equilibrium exists. First, however, we need to address to problem of solving the LP for a fixed pattern $\mathcal{B}$.

6.3 Column Generation

Note that in the pattern formulation of the previous section, as well as in the order 1, 2, and 3 formulation given above, there is an exponential number of variables (exponential in the number of items m). Clearly, the formulation itself cannot be kept in memory for even moderately-sized m. We can assume, on the other hand, that the number of constraints $|\mathcal{B}| + n + 1$ is manageable. A standard technique for solving such programs is *column generation*. In the simplex method, not all variables (columns) need to be kept in memory; in principle, at each iteration we only need to keep the basic variables in memory, and then identify new variables with appropriate reduced cost to enter the basis. The problem of finding variables with positive reduced cost is generally known as the "separation problem," and in our context it can be given a bidding interpretation: the variables that should enter the basis are precisely those that maximize the bidders' utilities at the given prices (i.e. dual solution). For the auctioneer, the variables that should enter the basis are those that correspond to revenue-maximizing partitions.

The connection between column generation and auction algorithms was first noted by de Vries and Vohra [32]. Here we elaborate further on this connection by describing the column generation method for the pattern formulation. We also give an alternate approach in which coefficients in the objective are incrementally increased, rather than set to their actual true values as soon as a column is generated. The advantage of the latter approach is that it does not require value queries, and so value information for any given bundle is only elicited incrementally from the bidders. The approach is also closely connected to the subgradient method and Parkes' *i*Bundle [78] as well as

Ausubel and Milgrom's ascending-proxy auction [7], as we discuss in Section 6.5.

Suppose we formulate the LP using a restricted set of columns and then solve it, which gives us prices p in the dual. A variable enters the basis if its reduced cost is positive, because the primal is a maximization problem. As we can see from the dual, the reduced cost of variable $y_i(S)$ is

$$v_i(S) - \sum_{T \in \mathcal{B}(S)} p(T) - \pi_i.$$

Let $y_i(S^*)$ be a basic variable. The reduced cost of a variable in the basis is 0. Hence, if the reduced cost of $y_i(S)$ is positive, we have

$$\begin{aligned} & v_i(S) - \sum_{T \in \mathcal{B}(S)} p(T) - \pi_i > 0 \\ \Rightarrow \quad & v_i(S) - \sum_{T \in \mathcal{B}(S)} p(T) - \pi_i > v_i(S^*) - \sum_{T \in \mathcal{B}(S^*)} p(T) - \pi_i \\ \Rightarrow \quad & v_i(S) - \sum_{T \in \mathcal{B}(S)} p(T) > v_i(S^*) - \sum_{T \in \mathcal{B}(S^*)} p(T) \end{aligned}$$

Therefore, to find a variable $y_i(S)$ to enter the basis, for each agent $i \in N$, we identify a bundle S^* such that $y_i(S^*)$ is a basic variable (if there is no such bundle then $\emptyset$ should be used because this is what the agent receives), and present it in a demand query to agent i along with the prices p. We can repeat this for every agent to find a set of at most n new columns. Note that the demand queries identify variables with maximal reduced cost.

Similarly, the reduced cost of variable $z(\omega)$ is

$$\sum_{S \in \omega} \sum_{T \in \mathcal{B}(S)} p(T) - \pi^s.$$

Let ω^* be a partition for which $z(\omega^*)$ is a basic variable (again, if there is no such variable the empty partition should be chosen, but this should never occur if agents

have nonzero valuations). Variable $z(\omega)$ is introduced only if its reduced cost is positive, which is the condition that

$$\begin{aligned} & \sum_{S \in \omega} \sum_{T \in \mathcal{B}(S)} p(T) - \pi^s > 0 \\ \Rightarrow & \sum_{S \in \omega} \sum_{T \in \mathcal{B}(S)} p(T) - \pi^s > \sum_{S \in \omega^*} \sum_{T \in \mathcal{B}(S)} p(T) - \pi^s \\ \Rightarrow & \sum_{S \in \omega} \sum_{T \in \mathcal{B}(S)} p(T) > \sum_{S \in \omega^*} \sum_{T \in \mathcal{B}(S)} p(T) \end{aligned}$$

To find a variable $z(\omega)$ to introduce, we find the partition ω that maximizes the auctioneer's revenue at prices p. Since the prices are given in a pseudo-additive representation and the pseudo-additive language is a bidding language, this problem can be formulated as an integer program (i.e. a winner determination MIP), as described in Section 4.2.5. Solving the program does not involve any queries to the agents.

The process of solving the primal LP via the simplex method with column generation can thus be given the following auction interpretation. At each round, we have a restricted set of columns. We solve the LP, obtaining solutions to the primal and dual with just these columns. We then identify a partition ω in our set of columns such that $z(\omega)$ is a basic variable. If ω does not maximize the auctioneer's revenue at prices p, we identify a partition that does maximize revenue and introduce it as a new column. This process is repeated until the basic solution found genuinely contains a revenue-maximizing partition, at which point we turn to the bidders. (The auctioneer and bidder phases alternate until a solution is reached.)

For each $i \in N$ we identify an S_i such that $y_i(S_i)$ is a basic variable in the solution to the primal. We present each S_i together with prices p, the solution to the dual,

as a demand query.[4] The interpretation is that the agent either accepts the bundle at the given prices, or else "bids" on another by replying with a different bundle. There are two variants at this point, that differ in how coefficients in the objective are updated.

1. We perform a value query on agent i's replied bundle to determine its column's coefficient in the objective.

2. We increase the coefficient on agent i's replied bundle by a fixed increment $\epsilon > 0$.

In the second variant, we maintain a coefficient of $\tilde{v}_i(S)$ in the objective for column $y_i(S)$, which throughout the auction satisfies

$$\tilde{v}_i(S) < v_i(S) + \epsilon. \tag{6.11}$$

The coefficients are set to 0 initially so the condition holds at the outset. We then use an ϵ-demand-query at each round to generate new columns. If the proposed bundle S^* is not accepted, then the replied bundle S must satisfy

$$\begin{aligned} v_i(S) - \sum_{T \in \mathcal{B}(S)} p(T) &> v_i(S^*) - \sum_{T \in \mathcal{B}(S^*)} p(T) + \epsilon \\ &> \tilde{v}_i(S^*) - \sum_{T \in \mathcal{B}(S^*)} p(T) \end{aligned}$$

and hence

$$v_i(S) > \tilde{v}_i(S^*) - \sum_{T \in \mathcal{B}(S^*)} p(T) + \sum_{T \in \mathcal{B}(S)} p(T).$$

[4]The allocation $(S_i)_{i \in N}$ may not be feasible, but this is of no consequence. The auction still proceeds correctly. The choice of S_i for each agent corresponds to a rounding of the fractional solution. The final solution to the LP will eventually be integer after enough constraints are generated, and in this case the corresponding allocation will be feasible.

It follows that if we set

$$\tilde{v}_i(S) = \tilde{v}_i(S^*) - \sum_{T \in \mathcal{B}(S^*)} p(T) + \sum_{T \in \mathcal{B}(S)} p(T) + \epsilon \tag{6.12}$$

then condition (6.11) is maintained. Also, note that because $y_i(S^*)$ was a basic variable, we originally had

$$\begin{aligned} & \tilde{v}_i(S) - \sum_{T \in \mathcal{B}(S)} p(T) \leq \tilde{v}_i(S^*) - \sum_{T \in \mathcal{B}(S^*)} p(T) \\ \Rightarrow \quad & \tilde{v}_i(S) \leq \tilde{v}_i(S^*) - \sum_{T \in \mathcal{B}(S^*)} p(T) + \sum_{T \in \mathcal{B}(S)} p(T) \end{aligned}$$

and comparing with (6.12) we see that the coefficient $\tilde{v}_i(S)$ has been strictly increased as $\epsilon > 0$.

In this second variant, the replied bundle S may already have a corresponding column in the LP formulation. In this case, only the coefficient is updated. Clearly, this gives us a finite procedure for solving the LP. Each time a bundle S is bid on its coefficient in the objective is increased. By condition (6.11), this can occur only a finite number of times. Otherwise, the algorithm proceeds just like the usual column-generation approach.

In column generation, various heuristics are possible to choose which columns should be retained in memory. At one extreme, every column generated can be retained for the remainder of the procedure; in order for this to work, we have to be certain that the LP will be solved before too many columns are generated. At the other extreme, only the current basic columns can be kept in memory, and the rest discarded. In this case care needs to be taken to avoid cycling, by using, for example, standard methods such as Bland's rule [14].

6.4 Cutting Planes

Once the LP formulation has been solved by column generation, the result may be a fractional solution. What we seek is an integer solution that defines a specific feasible allocation, together with CE prices that support it. As a step toward this, we need to strengthen the current LP formulation in order to eliminate the fractional solution from the feasible set. The process of strengthening the formulation and then solving it once again by column generation can then be repeated until an integer solution is obtained. A standard technique for strengthening the formulation is the use of *cutting planes* (see for example any of [9, 73, 109]). This involves introducing new constraints into the formulation that cut off a fractional solution from the feasible set, while leaving the integer solutions within the feasible set. Adding a constraint in the primal means an additional variable is introduced in the dual, which changes the price representation.

Again, de Vries and Vohra [32] were the first to note that cutting planes could be used to strengthen formulations for auction algorithms, with the interpretation that this increases the dimensionality of the price space. They suggested that column generation could be used within the context of a branch-and-price scheme [35] to give an auction protocol that dynamically updates the price representation, thus ensuring that a competitive equilibrium is reached. Here we show how extending the pattern that defines a pattern formulation can be interpreted as a cutting-plane method that strengthens the formulation as required.

We break the exposition into a series of lemmas. First we give a lemma about decompositions according to Algorithm 1. In words, it says that if T is a subset of S

such that no bundle in the decomposition of S clutters T (meaning that if T were to be considered next, it would become part of the decomposition for bundle S), then the bundles in $\mathcal{B}(S)$ that are subsets of T are exactly those in the decomposition $\mathcal{B}(T)$.

Lemma 9 *If $T \subseteq S$ and there is no $T' \in \mathcal{B}(S)$ such that $T' \diamond T$, then*

$$\mathcal{B}(T) = \{T' \in \mathcal{B}(S) \mid T' \subseteq S\}.$$

Proof. Suppose the first k elements $T' \subseteq T$ that are considered are selected or rejected identically in the decompositions of T and S. Clearly this is true when $k = 0$. If the next element $T' \subseteq T$ is rejected for the decomposition of T, this means that some element in T's decomposition so far clutters T'. As this element also appears in the current decomposition of S, T' is also rejected for this decomposition.

If the next element T' is accepted for the decomposition of T but not S, then there must be some $S' \subseteq S$ that clutters T'. Clearly S' must be ranked lower than T'. Hence we cannot have $S' \subseteq T$ because otherwise it would have cluttered T' for the decomposition of T as well. Since $S' \diamond T'$ and $T' \subseteq T$, we have $S' \cap T \neq \emptyset$. But because $S' \not\subseteq T$, we then have $S' \diamond T$. But this contradicts the fact that in the final decomposition of S, no bundle clutters T. Hence element T' is also rejected or accepted identically for the decompositions of both S and T.

Because no element in $\mathcal{B}(S)$ clutters T, the decomposition contains no element that is a strict superset of T. This, combined with the fact that every bundle considered for the decomposition of T is accepted or rejected identically as for the decomposition of S, proves the lemma. □

In the lemmas that follow, $\mathcal{B}$ is the original pattern, and $\bar{\mathcal{B}}$ is the pattern obtained by appending bundle T to $\mathcal{B}$, where $T \notin \mathcal{B}$.

Lemma 10 *If $T' \notin \mathcal{B}(T)$, then the constraint corresponding to $T' \in \mathcal{B}$ in the pattern $\mathcal{B}$ formulation also appears in the pattern $\bar{\mathcal{B}}$ formulation.*

Proof. There are three cases.

1. $T' \diamond T$. Then we have $\mathcal{B}(S) = \bar{\mathcal{B}}(S)$ for each $S \in \mathcal{B}^{-}1(T')$, because T cannot be added to these bundles' decompositions. Also, T' cannot be removed from their decompositions. As a result, the constraint corresponding to T' also appears in the pattern formulation with pattern $\bar{\mathcal{B}}$.

2. $T' \cap T = \emptyset$. In this case, T' remains in the decomposition of each $S \in \mathcal{B}^{-1}(T')$ whether T is added to the decomposition or not. Hence the constraint corresponding to T' remains intact after we append T to $\mathcal{B}$.

3. $T' \subseteq T$. In this case, adding T cannot change the decomposition of any $S \in \mathcal{B}^{-1}(T')$, because this would contradict Lemma 9. Any $S \in \mathcal{B}^{-1}(T')$ has T' in its decomposition, and if $T' \notin \mathcal{B}(T)$, then the conclusion of the lemma does not hold. This means some element of $\mathcal{B}(S)$ must clutter T.

□

The previous lemma identified which constraints remain intact in the formulation after adding T to the pattern. The next two lemmas show that the remaining constraints are broken up into two distinct constraints when moving to pattern $\bar{\mathcal{B}}$.

Lemma 11 *If $T' \in \mathcal{B}(T)$, then $S \in \mathcal{B}^{-1}(T')$ if and only if $S \in \bar{\mathcal{B}}^{-1}(T')$ or $S \in \bar{\mathcal{B}}^{-1}(T)$.*

Proof. Assume $S \in \mathcal{B}^{-1}(T')$. If T can be added to the decomposition of S, then $S \in \bar{\mathcal{B}}^{-1}(T)$. Otherwise T' remains in the decomposition of S. This proves the forward direction.

If $S \in \bar{\mathcal{B}}^{-1}(T')$, then we must have had $S \in \mathcal{B}^{-1}(T')$ because T' cannot become part of a bundle's decomposition by adding T when it was not there already. If $S \in \bar{\mathcal{B}}^{-1}(T)$, then we could add T to the decomposition of bundle S. By Lemma 9, this means all bundles in $\mathcal{B}(T)$ were originally in the decomposition of S, in particular T'. Hence $S \in \mathcal{B}^{-1}(T')$. This proves the reverse direction. □

Lemma 12 *If $T' \in \mathcal{B}(T)$, then $\{S \mid S \in \bar{\mathcal{B}}^{-1}(T')\} \cap \{S \mid S \in \bar{\mathcal{B}}^{-1}(T)\} = \emptyset$.*

Proof. $T' \in \mathcal{B}(T)$ implies $T' \subseteq T$, so T and T' cannot both appear in the decomposition of the same bundle. □

We are now ready to prove our main result. The idea is that all the constraints in the original formulation with pattern $\mathcal{B}$ are either present or implied by the constraints in the new formulation $\bar{\mathcal{B}}$.

Theorem 17 *Appending a bundle to a pattern leads to a stronger pattern formulation than the original.*

Proof. Note that the two formulations share the same variables, and share constraints (6.7) and (6.8). By Lemma 10, those constraints (6.6) that correspond to $T' \in \mathcal{B}$ such that $T' \notin \mathcal{B}(T)$ also appear in the formulation corresponding to the new pattern $\bar{\mathcal{B}}$. By Lemmas 11 and 12, the constraint corresponding to $T' \in \mathcal{B}$ such that $T' \in \mathcal{B}(T)$ in the original formulation is the sum of the constraints corresponding to

T' and T in the enw formulation. Hence every constraint in the pattern $\mathcal{B}$ formulation are either present or implied in the pattern $\bar{\mathcal{B}}$ formulation. □

Given these results, we see that adding a bundle T to a pattern is equivalent to adding a set of new constraints to the formulation, namely

$$\sum_{S\in\mathcal{B}^{-1}(T')}\sum_{i\in N} y_i(S) \le \sum_{\omega:\omega\cap\mathcal{B}^{-1}(T')\neq\emptyset} z(\omega) \quad (T'\in\mathcal{B}(T))$$
$$\sum_{S\in\mathcal{B}^{-1}(T)}\sum_{i\in N} y_i(S) \le \sum_{\omega:\omega\cap\mathcal{B}^{-1}(T)\neq\emptyset} z(\omega)$$

The constraints corresponding to $T'\in\mathcal{B}(T)$ in the original formulation become redundant and can be dropped.

We now assume that agents have superadditive valuations, so that order 2 CE prices exist. By the results of Section 3.3, it is sufficient to consider the following formulation.

$$\begin{aligned}
\max_{y(S),z(\omega)} \quad & \sum_{S\subseteq M} v(S)y(S) & \\
\text{s.t.} \quad & \sum_{S\in\mathcal{B}^{-1}(T)} y(S) \le \sum_{\omega:\omega\cap\mathcal{B}^{-1}(T)\neq\emptyset} z(\omega) & (T\in\mathcal{B}) \\
& \sum_{S\subseteq M} y(S) \le n & \\
& \sum_{\omega\in\Omega} z(\omega) \le 1 & \\
& y(S)\ge 0 & (S\subseteq M) \\
& z(\omega)\ge 0 & (\omega\in\Omega)
\end{aligned}$$

Here $y(S)$ is positive if some agents obtains S, and

$$v(S) = \max_{i\in N} v_i(S).$$

A fractional solution to the original program gives a fractional solution to the latter program, whereas given an integer solution to the latter program, we can derive an integer solution to the original program. To do this, if $y(S) = 1$, then we allocate S to the agent i that maximizes $v_i(S)$ (in case of ties we would need to solve a bipartite matching problem). This is correct by the results of Section 3.3.

The following result explains how to cut off a fractional solution (y^*, z^*).

Proposition 6 *Let (y^*, z^*) be an extreme point fractional solution for the formulation corresponding to pattern $\mathcal{B}$. If we append the set of bundles*

$$\mathcal{Q} = \{S \mid y^*(S) > 0\} \cup \{S \mid S \in \omega \text{ for some } \omega \text{ s.t. } z^*(w) > 0\}$$

to $\mathcal{B}$, in any order, then the formulation is strengthened and (y^*, z^*) is no longer feasible.

Proof. Suppose we append the bundles from $\mathcal{Q}$ to pattern $\mathcal{B}$, in any order. By Theorem 17, this strengthens the formulation. Note now that the decomposition of any bundle S such that $y^*(S) > 0$ is $\{S\}$ itself. Also, the decomposition of any $S \in \omega$ such that $z^*(\omega) > 0$ is again $\{S\}$ itself. Now assume that (y^*, z^*) is still a feasible extreme point solution to the resulting formulation. Since the formulation was strengthened, the solution must still be optimal. But note that there is now a unique positive $y*(S)$ variable on the left-hand side of any constraint, as well as a unique positive $z^*(\omega)$ variable on the right-hand side of any constraint. This means that y^* represents a convex combination of partitions, and since (y^*, z^*) was optimal, each of these partitions must also be optimal. But this contradicts the fact that (y^*, z^*) was an extreme point solution to the original program. □

To summarize, if we obtain a fractional extreme point solution (y^*, z^*), then appending the bundles in $\mathcal{Q}$ to the pattern cuts off the solution while strengthening the formulation. The simplex method, with or without column generation, always finds an extreme point solution. Note that the bundles in $\mathcal{Q}$ do not have to be added all at once. We can add them one at a time, each time checking whether the fractional solution is still feasible. In the worst-case all bundles will be added, thus ensuring that the fractional solution is cut off. The size of $\mathcal{Q}$ is at most the number of basic variables in the feasible solution, which is at most the number of constraints $|\mathcal{B}| + n + 1$. Also, note that we have the freedom to use various heuristics to choose the order in which bundles from $\mathcal{Q}$ are appended to $\mathcal{B}$. For instance, we may favor smaller or larger bundles, depending on any insights we may have on the structure of the problem.

6.5 Discussion

In this chapter, we gave a distributed algorithm for the CAP based on column and constraint generation. A flowchart of the combined process is given in Figure 6.2. Because the algorithm interacts with agents via demand queries, it has a clear auction interpretation: at each round, an allocation and prices are computed; these are presented to the agents, who bid on something else if they are not satisfied; the process repeats. The auction expands the expressiveness of the price representation incrementally. The auction adds nonlinearities into the price space as the rounds progress in order to ensure that a CE exists. This gives us prices that are in some sense intermediate between order 1 and 2 prices. Parkes' *i*Bundle(d) auction [78] uses an approach that is similar in spirit: the auction moves from order 2 to order 3

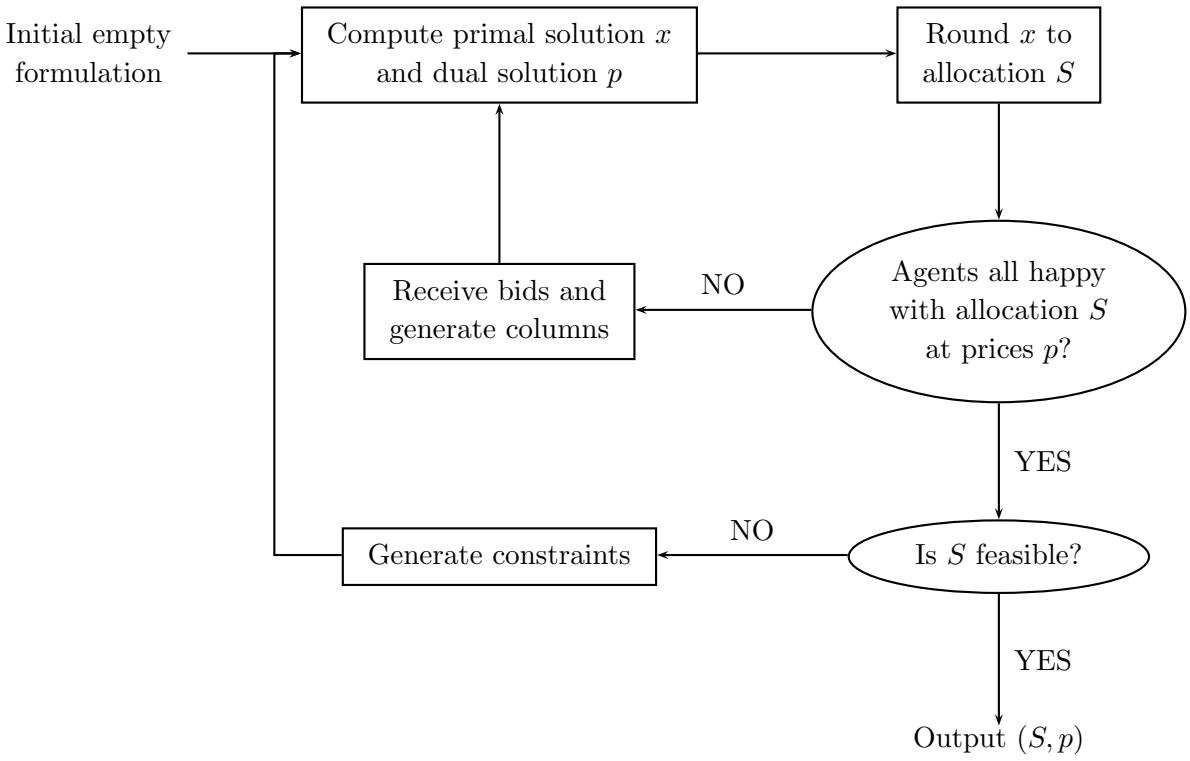


Figure 6.2: Flowchart of the column and constraint generation procedures.

prices gradually, introducing price discrimination for individual agents when this is necessary to ensure the existence of CE prices. We restricted ourselves to the case of superadditive valuations to focus on the issue of nonlinearity in pricing rather than discrimination. To handle general valuations, we would need an auction that handles both aspects, and this is an important next step for future work. In principle, the cutting-plane approach should apply to the general case as well. The question is how to detect that discrimination should be introduced rather than nonlinearity when we are faced with a fractional solution.

The column-generation approach explained here in detail is closely related to the subgradient method and *i*Bundle(3) [78] (equivalently, the ascending-proxy auction [7]) in particular. If we use the column-generation approach with the order 3 formulation from the outset, we recover a protocol that is extremely similar to *i*Bundle. In our design, the LP coefficients can be construed as the agents' bids. The prices (dual solutions) may or may not agree with these bids, so prices and bids are decoupled. If we use the order 3 formulation from the outset, however, we can simply use the current coefficients as order 3 CE prices, making the prices equal to the bids as in *i*Bundle. Note that with the order 3 formulation, no cutting planes are ever required. The only remaining difference is then that with *i*Bundle, entire best-response sets are provided at each round, whereas in our design just a single element of each set is needed. Moreover, our design would also work if entire best-response sets were provided: we could simply generate a new column for some or all of the bundles in the set. Conversely, it is simple to adapt *i*Bundle so that less than the full best-reponse set is required: revelation of a best-response set can simply be spread out over several rounds [7].

In the worst-case, our auction may start with linear prices, and end up with a formulation where prices are completely nonlinear. It is important to stress that in this case, the final formulation is not the same as the formulation used for *i*Bundle, so the two auctions always remain distinct. Prices remain pseudo-additive throughout, and this is a different language than XOR. The key difference is that in the pseudo-additive language, a bundle's decomposition can be computed without considering any bundle's value. In contrast, the atomic bid that gives a bundle its value in the

XOR language is precisely the subset that has highest value. Essentially, valuations succinctly represented in the pseudo-additive language can be viewed as generalizations of additive valuations, whereas valuations succinctly represented in the XOR language can be viewed as generalizations of unit-demand valuations.

Chapter 7

Application: Internet Advertising

Advertising has emerged as a dominant business model for Internet services, accounting for a significant portion of the revenue of companies such as Google and Yahoo. The two main forms of online advertising are *sponsored search* and *display advertising*. Sponsored search is the practice of listing advertisements alongside search results. Display advertising refers to the placement of banner advertisements on various websites.

In sponsored search, advertisers bid for placement on the page in an auction-style format where the larger their bid the more likely their listing will appear above other advertisements on the page. By convention, sponsored search advertisers generally bid and pay *per click*, meaning that they pay only when a user clicks on their advertisement, and do not pay if their advertisement is displayed but not clicked. The advertisements appear in a separate "sponsored" section of the page above or to the right of the algorithmic results. The sponsored results are displayed in a format similar to algorithmic results: as a list of items each containing a title, a text description,

and a hyperlink to a web page. Auctions in sponsored search are commonly called 'slot auctions' or 'position auctions' (we adopt the latter term).

In display advertising, it is still common to use a posted price, which varies depending on the website and also the size of the banner advertisement. Google, Yahoo, and MSN also provide matching functions that will display banners throughout websites in a network of affiliates. Here the matching is based on the content of the advertisement, and a bid provided by the advertiser. Advertisers generally pay *per impression*, meaning that they pay each time their banner is viewed. The trend in display advertising is to allow advertisers to specify what kinds of users they are targeting, e.g. by geographic location, interests, and even online behavior. This points to the need for auctions that can allow bidders to express complicated, even combinatorial, valuations over user attributes.

This chapter first surveys the academic literature on sponsored search, covering the relevant solution concepts for position auctions as well as the questions of efficiency and revenue under these concepts. It then turns to the problem of allocating advertisements among different websites, i.e. display advertising. I propose an expressive auction design for this environment. The core of the design is a bidding language that allows bidders to specify values for websites according to the sites' attributes (e.g. topic or geographic location of user), and also to place volume constraints on impressions to control exposure. The bidding language allows for fast demand queries, as well as fast allocation and pricing algorithms. I also propose a bidder feedback mechanism that can recommend bid increases to agents who would like to achieve greater volume. The expressive auction design is not an instance of the preference

elicitation framework given in earlier chapters, but shares the same core design principle: providing an intuitive interface for bidders while representing valuations in a way that allows for effective allocation and pricing algorithms.

Section 7.1 surveys the model, solution concepts, and central results on efficiency and revenue for position auctions in sponsored search. Section 7.2 then turns to the model for display advertising. Section 7.3 describes a bidding language for this domain, analyzing the kinds of valuations it describes, and giving the complexity of value and demand queries on instances of the language. Sections 7.4 and 7.5 discuss various methodologies for allocation and pricing, respectively. Section 7.6 addresses the incentives bidders would have to adjust bids. Section 7.7 describes the bidder feedback mechanism.

7.1 Sponsored Search

For sponsored search, we focus on the problem of allocating the slots next to search results for a single keyword among the advertisers. There are m positions to be allocated among n bidders. It is common to assume that the (expected) click-through rate of bidder i in position j is of the form $\alpha_i\gamma_j$, i.e. separable into an advertiser effect $\alpha_i \in [0,1]$ and position effect $\gamma_j \in [0,1]$. We have $\gamma_1 > \gamma_2 > \ldots > \gamma_k > 0$ and let $\gamma_j = 0$ for $j > k$. We will sometimes refer to α_i as the *relevance* of bidder i.

Bidder i has value v_i for each click. Bidders have quasi-linear utility, so that the utility to bidder i of obtaining position j at a price of p per click is

$$\alpha_i\gamma_j(v_i - p).$$

The auctioneer observes the advertiser effects, but the bidders' values remain private.

A weight w_i is associated with agent i, and agents bid for position. If agent i bids b_i, his corresponding *reported score*, or simply his *score*, is $s_i = w_i b_i$. His *true score* is $r_i = w_i v_i$. Agents are ranked by score, so that the agent with highest score is ranked first, and so on. Note that the weights may depend on the advertiser effects, but not on the bidder values, because the latter remain unobservable.

For clarity of notation, in this section agents are numbered such that agent i obtains position i, unless mentioned otherwise. An agent pays per click the lowest bid necessary to retain his position, so that the agent in position j pays $\frac{w_{j+1}}{w_j} b_{j+1}$. We refer to this payment rule as "second pricing."

The second-price payment rule is reminiscent of the second-price (Vickrey) auction used for selling a single item. Recall that in a Vickrey auction it is a dominant strategy for a bidder to reveal his true value for the item [106]. However, it is well-known that using a second-price rule in a position auction does not yield an incentive-compatible mechanism, either in dominant strategies or *ex post* Nash equilibrium [1, 57]. With second-pricing, there is no incentive for a bidder to bid higher than his true value per click, but there may be an incentive to shade the true value.

Given that second pricing is not strategy-proof, it is natural to ask whether there exists a payment rule that, together with a given weighting scheme, makes it a dominant strategy for the agents to bid their true values. If we rank agents by score, then charging agent i a total payment of

$$\sum_{j=i+1}^{n} (\alpha_i \gamma_{j-1} - \alpha_i \gamma_j) \frac{w_j b_j}{w_i} \tag{7.1}$$

makes truthful reporting a dominant strategy. To reduce this to a payment per click,

simply divide by $\alpha_i\gamma_i$. Lahaie [57] applied Holmstrom's Lemma [46] to derive the strategyproof payment rules for models of Yahoo and Google's position auctions, which coincide with the special cases where $w_i = 1$ and $w_i = \alpha_i$, respectively. Iyengar and Kumar [47] independently derived these rules using similar techniques. Aggarwal et al. [1] derived payment rule (7.1) and confirmed uniqueness from first principles, which gives some economic insight into why the rule works.

In typical position auctions such as those run by Yahoo and Google, bidders can adjust their bids up or down at any time. As Börgers et al. [16], Edelman et al. [33], and Varian [105] have noted, this can be viewed as a continuous-time process in which bidders constantly readjust their bids to obtain the position that gives them the highest surplus. If the process stabilizes the result can then be modeled as a Nash equilibrium in pure strategies of the static one-shot game of complete information, since each bidder will be playing a best-response to the others' bids.

To simplify the statement of results, for the remainder of this section we assume that there are as many positions as bidders. In a Nash equilibrium, each agent prefers his own position to the others given the bids, so the following inequalities must be satisfied:

$$\pi_i = \alpha_i\gamma_i(r_i - s_{i+1}) \quad (i \in N) \tag{7.2}$$

$$\pi_i \geq \alpha_i\gamma_j(r_i - s_{j+1}) \quad (i \in N, j > i) \tag{7.3}$$

$$\pi_i \geq \alpha_i\gamma_j(r_i - s_j) \quad (i \in N, j < i) \tag{7.4}$$

Here π_i can be interpreted as the agent in position i's weighted utility. Agent i's weighted utility for position j at price b per click is defined simply as $w_i\alpha_i\gamma_j(v_i - b)$.

Börgers et al. [16] show that there can be a multitude of pure-strategy Nash

equilibria in a position auction. To get an idea of the allocations that can arise in Nash equilibrium, we can ask, given an allocation of positions to bidders together with bidder values, whether there exist a vector of scores s such that inequalities (7.2)–(7.4) are satisfied. The question can be answered using linear programming methods to test for the feasibility of the inequalities (7.2)–(7.4). Lahaie [57] also gives some simple necessary conditions for the answer to be affirmative.

The multiplicity of possible equilibrium allocations makes it difficult to provide precise results on efficiency or revenue in Nash equilibrium. Varian [105] introduced a refinement of the Nash equilibrium concept for position auctions which he called "symmetric equilibrium." Edelman et al. [33] independently introduced this refinement and called it "locally envy-free equilibrium." With a slight modification we can make the Nash equilibrium inequalities above resemble those that arise in the assignment problem. In inequalities (7.4), we replace s_j by s_{j+1}. For clarity of notation we let $p_i = s_{i+1}$. A *symmetric* NE then satisfies

$$\pi_i = \alpha_i \gamma_i (r_i - p_i) \quad (i \in N) \tag{7.5}$$

$$\pi_i \geq \alpha_i \gamma_j (r_i - p_j) \quad (i \in N, j \neq i) \tag{7.6}$$

Varian [105] shows that symmetric equilibrium is indeed a refinement of Nash equilibrium, and that a symmetric equilibrium always exists. In addition, in symmetric equilibrium, agents are necessarily ranked by true score.

The set of symmetric equilibria forms a lattice [101, 105]. In particular, it has minimal and maximal elements, which minimize and maximize the auctioneer's revenue among the set of symmetric equilibria, respectively. The maximal and minimal elements have simple closed-form expressions, namely the upper- and lower-recursive

solutions for symmetric equilibria given by Varian [105] and Edelman et al. [33]. The form of the minimal element is particularly interesting:

$$\sum_{j=i+1}^{n} \alpha_i(\gamma_{j-1} - \gamma_j)\frac{w_j v_j}{w_i}. \tag{7.7}$$

(See Varian [105] or Lahaie and Pennock [60].) Note that payment (7.7) agrees exactly with (7.1). This is not an accident: the set of symmetric equilibria coincides with the set of dual solutions to the linear programming formulation of the position auction as an assignment problem, and Leonard [62] has shown that the minimal dual solutions in the assignment problem coincide with Vickrey payments. Hence minimal symmetric equilibrium bids should coincide with Vickrey payments, or, in the case of a position auction, the weighted equivalents of Vickrey payments, which are given by Holmstrom's lemma. As a result, minimal symmetric equilibria and strategy-proof position auctions are revenue-equivalent, a point first made by Aggarwal et al. [1].

We now turn to the question of efficiency under each of these solution concepts: dominant strategies, Nash equilibrium, and symmetric equilibrium. To maximize total value, we need to order the agents according to some permutation σ such that the inner product of the vectors $(\alpha_{\sigma(j)}v_{\sigma(j)})_{j\in K}$ and $(\gamma_j)_{j\in K}$ is maximized. Because $\gamma_1 > \gamma_2 > \ldots > \gamma_k$, it is efficient to rank agents in decreasing order of $\alpha_i v_i$, by standard results on rearrangements (see Hardy et al. [44] and also Lahaie [57]).

When using the strategy-proof payment rule (7.1), agents reveal their true values and therefore are ranked by true score. Hence, if we take $w_i = \alpha_i$, the resulting equilibrium allocation is efficient. This fact applies to the symmetric equilibrium concept as well, because in symmetric equilibrium agents are ranked by true score. As for Nash equilibrium, Lahaie [57] provides a constant-factor bound on the possible

deviation from efficiency. We denote the total value of an allocation σ of positions to agents by $\kappa(\sigma) = \sum_{j=1}^{k} \gamma_j r_{\sigma(j)}$. Let

$$L = \min_{j=1,\ldots,k-1} \min \left\{ \frac{\gamma_{j+1}}{\gamma_j}, 1 - \frac{\gamma_{j+2}}{\gamma_{j+1}} \right\}$$

Let η be the permutation such that $r_{\eta(1)} \geq \ldots \geq r_{\eta(k)}$. The bound is as follows.

Proposition 7 *For an allocation σ that results from a pure-strategy Nash equilibrium of a position auction, we have $\kappa(\sigma) \geq L\kappa(\eta)$.*

For the common exponential decay model of $\gamma_i = \delta^{1-i}$ for $\delta > 1$, the factor becomes $L = \min\{\frac{1}{\delta}, 1 - \frac{1}{\delta}\}$. Feng et al. [37] report that an exponential decay model with $\delta = 1.428$ fits their Overture click-through rate data well. In this case, $L \approx 1/3.34$. Though being a factor of more than 3 away from the efficient value may seem unacceptable, we stress that this is a worst-case bound, and we would expect actual deviations to be much less than this in practice.

As for revenue, again, because multiple different allocations can arise in Nash equilibrium, it is difficult to give a precise characterization of equilibrium revenue. Revenue in minimal symmetric equilibrium is amenable to analysis, on the other hand, and this selection is also relevant because it provides a lower bound on the revenue in any symmetric equilibrium. Recall also that dominant-strategy equilibrium is revenue-equivalent to minimal symmetric equilibrium in position auctions, as explained above.

Lahaie and Pennock [60] provide a mathematical program whose solution gives the revenue-optimal ranking rule for a position auction given a distribution F over bidder values and relevance (with corresponding density f), assuming the minimal symmet-

ric equilibrium is played. According to their analysis, bidders should be ranked by "virtual score" $\alpha_i \psi(\alpha_i, v_i)$ to optimize revenue, where

$$\psi_i(\alpha_i, v_i) = v_i - \frac{1 - F_i(v_i|\alpha_i)}{f_i(v_i|\alpha_i)}.$$

However, we are constrained to ranking rules that correspond to a certain weighting scheme $w_i \equiv g(\alpha_i)$. In general it is not possible to reproduce virtual valuations via a weighting scheme, but a revenue-optimal ranking rule can be implemented in certain special cases, e.g. if the bidders' values are distributed uniformly and uncorrelated with relevance.

7.2 Display Advertising

We now turn to the problem of allocating advertisements across different sites, rather than within a single webpage, and adapt the model accordingly. The model we give is best suited to the domain of online display advertising, rather than sponsored search. Let N be the set of bidders and let M be the set of sites. Let $n = |N|$ and $m = |M|$ be the numbers of bidders and sites, respectively. We assume that each site has exactly one position available for an advertisement.

We denote the set of real-valued vectors with entries indexed by elements of M by $\mathbf{R}^M$. The analogous set of integer-valued vectors is denoted by $\mathbf{Z}^M$. A non-negative element of $\mathbf{Z}^M$ represents a set of impressions on the various sites. Each bidder i has a valuation defined over sets of impressions, $v_i : \mathbf{Z}_+^M \to \mathbf{R}_+$. We consider valuations over impressions because the display advertising domain typically prices impressions rather than clicks; the model could be easily adapted to per-click pricing

by introducing advertiser effects as in Section 7.1.

We will only need to consider linear prices, so prices are elements of $\mathbf{R}^M$. (The reason for this restriction is that linear competitive equilibrium prices will always exist in our model, as explained in Section 7.3.2.) For $x \in \mathbf{Z}^M$, we denote the entry corresponding to site j by $x(j)$, and similarly for elements of $\mathbf{R}^M$. The total price of a set of impressions $x \in \mathbf{Z}_+^M$ at prices $p \in \mathbf{R}^M$ is the inner product $\langle p, x \rangle$, where

$$\langle p, x \rangle = \sum_{j \in M} p(j)x(j).$$

Bidders have quasi-linear utilities, so that the utility to bidder i of impressions $x \in \mathbf{Z}_+^M$ at prices $p \in \mathbf{R}^M$ is $v_i(x) - \langle p, x \rangle$.

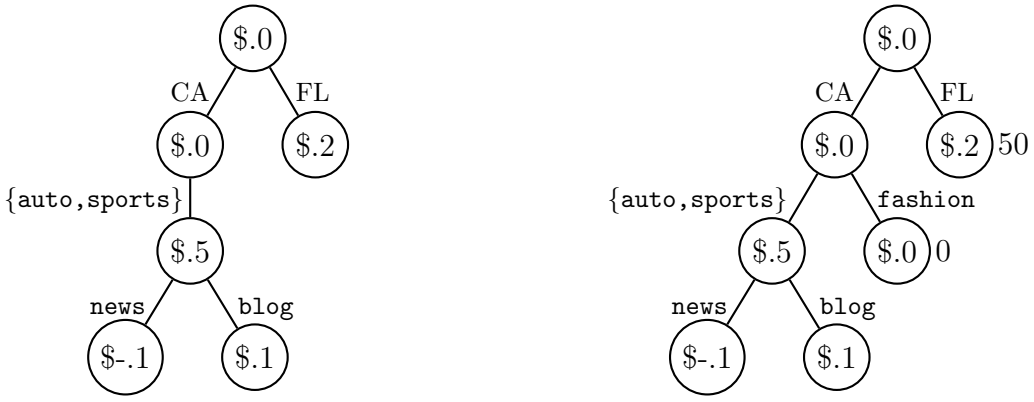
7.3 Bidding Language

We first describe the bidding language in terms of the data structures that would be used to encode its instances, and then turn to the properties of the valuations it describes.

7.3.1 Bid Trees

To encode valuations, we propose "bid trees" that enable advertisers to specify values for various kinds of impressions, where impressions are differentiated according to certain attributes (e.g. geographic location).[1] The attributes and their possible values are determined by the auctioneer.

[1]Tree-based languages have been proposed for combinatorial auctions, e.g. the $\mathcal{L}_{GB}$ language of Boutilier and Hoos [20], as well as for combinatorial exchanges, e.g. TBBL by Cavallo et al. [22]. The language proposed here represents a smaller class of valuations than either of those languages, but is specially tailored to the domain of display advertising.



(a) Bid tree.

(b) Capacitated bid tree.

Figure 7.1: Instances of the bidding language for keyword auctions.

For instance, suppose a car manufacturer wishes to run an online campaign to advertise a new truck model. The campaign should only run in California and Florida. In Florida, the campaign should have a limited exploratory run across a variety of sites, to see what demographics respond best to the new model. In California, the campaign should project a "rugged" image for the truck. The company therefore decides to run its banner ad next to content with an "automotive" or "sports" theme, and to avoid any content with a "fashion" theme. It also decides to value exposure on blogs and devalue mainstream news sites, because it would like the campaign to have "grassroots" appeal.

A candidate bid tree for this kind of valuation is given in Figure 7.1(a). The root node represents all impressions over all sites in the publishing network. The value of an impression from a certain source is evaluated by traveling down the tree following attributes that apply, starting at the root, and summing the values in the nodes along

the way. (A bidder may branch on a set of attributes for succinctness.) Here, any impression that is not from CA or FL has value \$0. In FL, impressions have value \$.2. In CA, impressions have value \$.5, but only if they are from automotive or sports sites. Finally, impressions from automotive or sports sites in CA are valued \$.1 more if they are on blogs, but \$.1 less if they are on news sites.

However, we argue that there is still a disconnect between the bid tree in Figure 7.1(a) and the company's valuation. Exposure outside of CA or FL is worthless, but does not do any harm if it occurs, so it indeed has value \$0. On the other hand, exposure on fashion sites in CA does do harm, because it goes against the brand image the advertiser is trying to project in that state. Also, there is no telling how many impressions will come from FL, even though the run there should be limited. Hence we further allow advertisers to annotate nodes with capacities. In Figure 7.1(b), there is a new node for fashion sites in CA, with a capacity of 0 to ensure no impressions. The FL node now has a capacity of 50,000 to ensure the campaign there is limited to this amount of exposure.[2]

7.3.2 Properties

The value functions encoded by bid trees can be described formally as follows. In the site listing M, sites that are identical across all attributes may be considered equivalent, and their available impressions can be aggregated. The auctioneer decides which combination of attributes each site exhibits, and makes the categorizations

[2]A capacity of 50,000 also ensures that the advertising budget in FL will not exceed \$10,000 because the auction design will never charge an advertiser more than its value per impression. Capacity and budget constraints are not equivalent, however. With a budget constraint, the effective volume constaint varies as prices change. I will not handle budget constraints of this form.

public to the bidders.

Given a bid tree, we can define a family $\mathcal{T}$ of subsets of M corresponding to each node. For instance, for the tree of Figure 7.1 we would have the set of all sites that appear in CA, the set of all automitive or sports sites in CA, the set of all automotive or sports blogs in CA, etc. It is straightforward to see that, because of the tree format, this family is *laminar*; i.e. for any $T, T' \in \mathcal{T}$, either $T \subseteq T'$, $T' \subseteq T$, or $T \cap T' = \emptyset$. To each $T \in \mathcal{T}$ is associated an integral capacity c_T (possibly $+\infty$) and a bid b_T. Define the indicator functions

$$v_T(r) = \begin{cases} b_T r & \text{if } 0 \leq r \leq c_T \\ -\infty & \text{otherwise} \end{cases}$$

for each $T \in \mathcal{T}$. Here r is a non-negative scalar, representing a certain number of impressions.

The agent's value for a set of impressions $x \in \mathbf{Z}^M$ is then

$$v(x) = \sum_{T \in \mathcal{T}} v_T(x(T)) \tag{7.8}$$

where $x(T)$ is shorthand for $\sum_{j \in T} x(j)$, i.e. the sum of all impressions over all sites in T.

Danilov et al. [29] show that functions of the form (7.8), where $\mathcal{T}$ is laminar and each v_T exhibits decreasing marginal values over some interval (and is $-\infty$ outside this interval), satisfy the substitutes condition.[3] (In our case the marginal values are constant up to a threshold, not just weakly decreasing.) Recall from Section 3.3 that order 1 CE prices exist when the agents have substitutes valuations—this justifies

[3]Danilov et al. call functions that satisfy the substitutes condition "M$^\natural$-concave." The "substitutes condition" is terminology from the microeconomics literature.

our restriction to linear prices. However, the question of whether bidders would bid truthfully with an appropriate choice of order 1 CE prices remains—incentive concerns are addressed later in Section 7.6.

We chose bid trees as a language in part because the associated value functions satisfy the substitutes condition. Not only does the condition guarantee the existence of order 1 CE prices, it also allows for fast allocation and pricing algorithms. The bid tree language is indeed suitable for bidding, in the sense of Chapter 4. Let $y \in \mathbf{Z}^M$ be a set of impressions whose value is queried. The integer program to evaluate a value query is as follows.

$$\begin{aligned} \max \quad & \sum_{T \in \mathcal{T}} \sum_{j \in T} b_T y(j) \\ \text{s.t.} \quad & \sum_{j \in T} y(j) \leq c_T \quad (T \in \mathcal{T}) \end{aligned} \tag{7.9}$$

This program is somewhat artificial because it has no choice variables, but recall that when it is embedded within a winner-determination MIP, the y vector becomes a choice variable. The objective evaluates the total value of the impressions following the semantics of the bid tree. However, if any of the capacity constraints are violated, the program is infeasible, which is a flag that indicates the value is $-\infty$. When this formulation is embedded within a multi-agent allocation program, the auctioneer will never allocate sets of impressions that violate capacity constraints.

7.3.3 Queries

As explained in Chapter 4, there are two fundamental queries that are typically made on bidding languages in auction-style algorithms: value and demand queries.

Clearly, the time to evaluate the response to a value query is at most the depth of the bid tree. This is at most $t = |\mathcal{T}|$, the size of the bid tree, but can be around the order of $\log t$ if the tree is balanced.

For a valuation of the form (7.8) derived from a bid tree, the response to a demand query (given linear prices) can be computed using Algorithm 6, which is a greedy algorithm. For $j \in M$, χ_j denotes the unit vector with entry j being 1 and all others 0. Let $\mathbf{0}$ be the zero vector. Let $v_j = \sum_{T \ni j} b_T$ be the marginal value from an impression on site j, and let $\pi(j) = v_j - p(j)$ be the marginal surplus. (We can refer to the marginal value of an impression without reference to impressions that have already been obtained, because the marginal values described by a bid tree are always constant as long as volume constraints are not violated.)

Algorithm 6 computes a response to a demand query, given prices and a proposed set of impressions. It is a greedy algorithm. It considers impressions in decreasing order of marginal utility, and collects each impression considered unless the addition violates a volume constraint. After sorting the elements, the greedy algorithm runs in linear time for a worst-case bound of $O(m \log m + mt \log C)$, where t is the size of the bid tree and C the maximum capacity over all nodes in the tree.[4] We record the correctness of the algorithm as a lemma.

Lemma 13 *Algorithm 6 correctly outputs a set of impressions $x \in \arg\max v - p$, or* YES *if x_0 maximizes utility.*

Proof. Let x be a set of impressions with $v(x) > -\infty$. If $x(j) > 0$ but $\pi(j) < 0$,

[4]Murota [72] gives a steepest ascent algorithm that can compute the response to demand queries in time $O(tm^2C)$ for any valuation that satisfies the substitutes property (here t is the time required for an oracle to answer value queries). Bid tree valuations have a more special structure that allows for the faster, linear-time greedy algorithm given here.

Input: A set of impressions $x_0 \in \mathbf{Z}^M$ and linear prices $p \in \mathbf{R}^M$.

Output: A utility-maximizing set of impressions $x \in \mathbf{Z}^M$, or YES if x_0 maximizes utility.

Set $x := \mathbf{0}$.

For each $T \in \mathcal{T}$, set $d_T := c_T$.

Discard the elements $j \in M$ that have $\pi(j) < 0$.

Sort the remaining elements according to π. In case of a tie, break arbitrarily.

foreach $j \in M$ *in order* **do**
  Set $k := \min_{\{T \in \mathcal{T} \mid T \ni j\}} d_T$.
  Set $x := x + k\chi_j$.
  foreach $T \in \mathcal{T}$ *such that* $T \ni j$ **do**
    Set $d_T := d_T - k$.
  end
end

if $v(x_0) - \langle p, x_0 \rangle = v(x) - \langle p, x \rangle$ **then**
  Output YES.
end
else
  Ouput x.
end

Algorithm 6: Greedy algorithm for demand queries on bid trees.

then we can increase utility while respecting volume constraints by setting $x(j) := 0$. So we can safely restrict our attention to the set of sites j such that $\pi(j) \geq 0$, as Algorithm 6 does. Let U be the set of such sites. We consider the restriction of v_U of v to such sites. By Theorem 6.13 of Murota [72] this restriction still satisfies

the substitutes condition, and so does the function $v_U - p_U$, where p_U denotes the restriction of p to U.

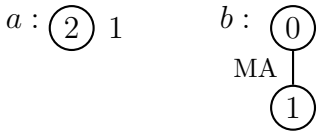
Since $v_U - p_U$ satisfies the substitutes condition, the set of x such that $v_U(x) - \langle p_U, x\rangle > -\infty$ is a matroid, by Proposition 6.1 of Murota [72]. Therefore, by a result of Edmonds [34] (see also Theorem 13.20 in Korte and Vygen [53]), the greedy algorithm correctly identifies a maximum weight basis, which in this case corresponds to a utility-maximizing bundle x. The final if-statement ensures that YES is returned if x_0 maximizes utility. □

Because our bidding language only represents substitutes valuations, demand queries with linear prices can be evaluated in polynomial time. Among the bidding languages introduced in Section 4.2, only XOR had this feature; for the others, evaluating a demand query is NP-hard [74]. The XOR language does not seem appropriate for this domain, however. It does not provide the ability to specify volume constraints, and cannot succinctly represent the additive valuation, which is plausible in this domain.

7.4 Allocation

A natural way to allocate impressions in an online fashion is to give each arriving impression to the advertiser who values it most among those advertisers for whom the extra impression would not violate any volume constraints. We note that this scheme is not efficient in our context. Volume constraint are the source of the complication, as the following example illustrates.

Example. There are two bidders $\{a, b\}$ and two impressions available, one from Massachusetts (MA) and one from California (CA). The valuations as bid trees are



If the MA impression comes before the CA impression, the greedy scheme assigns MA to a and CA to b, for a value of 2. But it is optimal to assign MA to b and CA to a, for a value of 3.

To better allocate impressions over time, we assume the auctioneer has an estimate of the number of impressions available (over some fixed time period) on each site. (Again, a site can be construed as a distinct combination of attributes, and we aggregate the impressions over sites that are identical over all attributes.) Let $z(j)$ be the average number of views received by site j.

The efficient allocation problem can be formulated as a linear program. We distinguish between the nodes, bids, and capacities of different bidders by indexing the variables by their corresponding agent i. The linear programming formulation is then

$$\begin{aligned}
\max_{x} \quad & \sum_{i \in N} \sum_{T \in \mathcal{T}_i} \sum_{j \in T} b_{iT} x_i(j) \\
\text{s.t.} \quad & \sum_{j \in T} x_i(j) \leq c_{iT} \qquad (T \in \mathcal{T}_i,\, i \in N) && (7.10) \\
& \sum_{i \in N} x_i(j) \leq z(j) \qquad (j \in M) && (7.11) \\
& x_i(j) \geq 0 \qquad (i \in N,\, j \in M)
\end{aligned}$$

where (7.10) enforces the bidder's capacity constraints and (7.11) ensures that the number of impressions does not exceed the supply.

Because the agents' valuations satisfy the substitutes condition, this linear program in fact has an integer optimal solution. In our context this is not important, because the z_s are estimates anyway, and presumably large. However, this fact does make the auction design suitable to situations where there is a small, fixed number of impressions available (e.g. in a television station's daily programming schedule), so we record it as a proposition.

Proposition 8 *When agents submit their valuations as bid trees, the corresponding allocation LP has an integer optimal solution.*

Proof. The primal is clearly feasible and bounded, and hence so is the dual. From Lemma 14 below, optimal primal and dual solutions x and (π, p) imply that (x, p) is a competitive equilibrium, where the allocation x is possibly fractional. So a fractional CE exists. Because the bidders all have substitutes valuations, if then follows from Theorem 11.14 of Murota [72] that there exists a CE (x', p) where x' is integer. Applying Lemma 14 once again, we see that x' is an integer optimal solution to the primal. □

There are several algorithms available to solve the allocation LP. Any variant of simplex method would likely be effective [9, 24]. In this case, though, there are also purely combinatorial algorithms, which may be easier to implement. Kelso and Crawford's [50] original ascending-price auction for substitutes valuations can be adapted to this setting, and leads to an approximately optimal solution. The approximation can be improved to the extent needed by decreasing the bid increment (which increases the runtime as well). Ausubel [5] gives an ascending-auction procedure that can terminate at the smallest or largest CE price vectors, although it involves solv-

ing a submodular function minimization at each round, which could be costly (this is polynomial time, but the worst-case exponents are very high—see Chapter 45 of Schrijver [100]). Since demand queries are quick to evaluate, the primal-dual method may hold the greatest promise if we need an exact solution.

7.5 Pricing

The linear programming approach to the allocation problem is useful because it also provides prices for various kinds of impressions. Let $p(s)$ be the dual variable corresponding to the constraint for s in (7.11) in the LP, and let π_{iT} be the dual variable corresponding to the constraint for i and $T \in \mathcal{T}_i$ in (7.10). The dual of the LP is as follows.

$$\begin{aligned}
\min_{\pi,p} \quad & \sum_{i\in N}\sum_{T\in\mathcal{T}_i} \pi_{iT} c_{iT} + \sum_{j\in M} p(j) z(j) & & \\
\text{s.t.} \quad & \sum_{\{T\in\mathcal{T}_i \,|\, T\ni j\}} \pi_{iT} \geq \sum_{\{T\in\mathcal{T}_i \,|\, T\ni j\}} b_{iT} - p(j) & (i\in N,\, j\in M) & \qquad (7.12)\\
& \pi_{iT} \geq 0 & (i\in N,\, T\in\mathcal{T}_i) & \\
& p(j) \geq 0 & (j\in M) &
\end{aligned}$$

We claim that for a dual solution (π, p), the p component represents competitive equilibrium prices. In this context, we have a CE if for any efficient allocation $(x_i^*)_{i\in N}$, we have for all $i \in N$ and $x_i \in \mathbf{R}^M$,

$$v_i(x_i^*) - \langle p, x_i^*\rangle \geq v_i(x_i) - \langle p, x_i\rangle,$$

and any j such that $\sum_{i\in N} x_i^*(j) < z(j)$ has $p(j) = 0$; this last condition ensures that the allocation maximizes the auctioneer's revenue. Competitive equilibrium prices

are also always non-negative.

Lemma 14 *There is a vector π such that x and (π, p) are optimal primal and dual solutions to the allocation LP if and only if (x, p) is a competitive equilibrium.*

Proof. Let x and (π, p) be optimal primal and dual solutions, respectively. By the dual constraints, $p \geq 0$. By complementary slackness, $p(j) > 0$ implies that $\sum_{i \in N} x_i(j) = z(j)$, and the contrapositive of this fact implies that any site j whose impressions are not fully allocated has $p(j) = 0$. This shows that the allocation x maximizes revenue for the seller.

By complementary slackness, if $x_i(j) > 0$ then

$$\sum_{\{T \in \mathcal{T}_i \,|\, T \ni j\}} \pi_{iT} = \sum_{\{T \in \mathcal{T}_i \,|\, T \ni j\}} b_{iT} - p(j).$$

Summing over all $j \in M$ yields

$$\sum_{j \in M} \sum_{\{T \in \mathcal{T}_i \,|\, T \ni j\}} \pi_{iT} x_i(j) = \sum_{j \in M} \sum_{\{T \in \mathcal{T}_i \,|\, T \ni j\}} b_{iT} x_i(j) - \sum_{j \in M} p(j) x_i(j). \tag{7.13}$$

The right-hand side of this equality is the surplus to bidder i from outcome (x, p). Let x' be any feasible allocation. Summing constraints (7.12) yields

$$\sum_{j \in M} \sum_{\{T \in \mathcal{T}_i \,|\, T \ni j\}} \pi_{iT} x'_i(j) \geq \sum_{j \in M} \sum_{\{T \in \mathcal{T}_i \,|\, T \ni j\}} b_{iT} x'_i(j) - \sum_{j \in M} p(j) x'_i(j). \tag{7.14}$$

The right-hand side of this inequality is the surplus to bidder i from outcome (x', p). The left-hand side of (7.13) can be re-written as

$$\sum_{j \in M} \sum_{\{T \in \mathcal{T}_i \,|\, T \ni j\}} \pi_{iT} x_i(j) = \sum_{T \in \mathcal{T}_i} \pi_{iT} \left(\sum_{j \in T} x_i(j) \right),$$

and the right-hand side of (7.14) can be rewritten analogously. By complementary slackness, $\pi_{iT} > 0$ implies that $\sum_{j \in T} x_i(j) = c_i T$, so the vector x maximizes the last

expression, given the primal feasibility constraints. Hence

$$\sum_{j \in M} \sum_{\{T \in \mathcal{T}_i \mid T \ni j\}} \pi_{iT} x_i(j) \geq \sum_{j \in M} \sum_{\{T \in \mathcal{T}_i \mid T \ni j\}} \pi_{iT} x_i'(j). \tag{7.15}$$

Combining (7.13), (7.14), and (7.15) shows that x maximizes bidder i's surplus at prices p. Hence (x, p) is a competitive equilibrium.

Now let (x, p) be a competitive equilibrium. We define a vector π by doing the following for each $i \in N$. First, set $\pi_{iT} = 0$ for each $T \in \mathcal{T}_i$ such that $\sum_{j \in T} x_i(j) < c_{iT}$. From the remaining $T \in \mathcal{T}_i$, for each $j \in M$ such that $x_i(j) > 0$, let T_j be the minimal element of $\mathcal{T}_i$ such that $T \ni j$. Because $\mathcal{T}_i$ is laminar, this choice is unique. The family $\{T_j\}_{j \in M}$ is also laminar. For each element T_j in this family, let $\bar{T}_j$ be the minimal element of the family that is a subset of T_j. Again, because of the laminar property, this element is either well-defined or does not exist. We set

$$\pi_{iT_j} = \sum_{\{T \in \mathcal{T}_i \mid T \ni j\}} b_{iT} - p_j - \pi_{i\bar{T}_j},$$

where if $\bar{T}_j$ does not exist, $\pi_{i\bar{T}_j} = 0$. This is a recursive definition: we can evaluate the π_i vector by visiting the elements of $\{T_j\}_{j \in M}$ from maximal to minimal. By construction, we have

$$\sum_{\{T \in \mathcal{T}_i \mid T \ni j\}} \pi_{iT} = \sum_{\{T \in \mathcal{T}_i \mid T \ni j\}} b_{iT} - p(j)$$

for each j such that $x_i(j) > 0$. If $x_i(j) = 0$, then let T_k be a minimal element from our family that contains j. If there is no such k, the surplus from each unit of j must be zero, because otherwise x would not maximize agent i's utility (we could take an additional unit of j without violating any volume constraints). Now, the marginal utility of j must be less than the marginal utility of k, because otherwise we could

switch one unit of k for one unit of j, which would strictly increase utility without violating any constraints, a contradiction. Hence, we have

$$\begin{aligned}\sum_{\{T\in\mathcal{T}_i \mid T\ni k\}} \pi_{iT} &= \sum_{\{T\in\mathcal{T}_i \mid T\ni j\}} \pi_{iT} \\ &= \sum_{\{T\in\mathcal{T}_i \mid T\ni j\}} b_{iT} - p(j) \\ &\geq \sum_{\{T\in\mathcal{T}_i \mid T\ni k\}} b_{iT} - p(j).\end{aligned}$$

So the constructed vector π is feasible. Finally, note that if $p(j) > 0$, we must have $\sum_{i\in N} x_i(j) = z_j$, or else the allocation would not maximize the seller's revenue. As x and (π, p) are feasible primal and dual solutions that satisfy the complementary slackness conditions, they are optimal. □

In fact, because each agent's valuation satisfies the substitutes condition, the set of competitive equilibrium prices is a lattice under the usual meet and join operations for real vectors, defined as

$$(p \vee q)(j) = \max\{p(j), q(j)\}$$

$$(p \wedge q)(j) = \min\{p(j), q(j)\}$$

for all $j \in M$. We record this as a proposition; it will be relevant when we discuss incentives later.

Proposition 9 *The set of competitive equilibrium prices is a lattice when valuations are described by bid trees.*

Proof. From Proposition 8 and Lemma 14, there exists a CE (x, p) where x is integer. Because the bidders' valuations all satisfy the substitutes condition, it follows from

Theorem 11.16 of Murota [72] that the set of CE prices forms a lattice (see also Gul and Stachetti [43]). □

The lattice property allows the auctioneer to be consistent in his choice of prices. Since a lattice has a unique minimal element $\underline{p}$ and a unique maximal element $\bar{p}$, the auctioneer may choose to consistently implement either of these. The minimal element gives the most possible surplus to the bidders, while the maximial element gives the most possible revenue to the seller, among the set of CE prices. Because the set of dual solutions to an LP is convex, so is the set of CE prices, and so $\alpha\underline{p}+(1-\alpha)\bar{p}$ is also a vector of CE prices, for any $\alpha \in [0,1]$. This allows the auctioneer to modulate the allocation of surplus between the bidders and seller.

To compute a maximal or minimal CE price vector, we replace the objective in the dual with

$$\sum_{j \in M} p(j)z(j)$$

i.e. it becomes simply the auctioneer's revenue. We also introduce the constraint

$$\sum_{i \in N} \sum_{T \in \mathcal{T}_i} \pi_{iT} c_{iT} + \sum_{j \in M} p(j)z(j) = \text{OPT}$$

where OPT is the optimal value of the primal program (or equivalently the original dual program, by strong duality). This ensures that the solution for the new program will be optimal for the original dual program. To compute minimal CE prices, we use a minimization in the objective, because the vector of CE prices that minimizes the auctioneer's revenue is the minimal element of the lattice. To compute the maximal element, we use a maximization in the objective. This approach to computing the maximal and minimal elements was suggested by Parkes [80], and it works for any

integral LP formulation of the CAP. In our case, because bid trees represent substitutes valuations, there also exist purely combinatorial algorithms; Chapter 12 of Murota [72] shows how the problem of computing either element can be formulated as the dual of a shortest path problem.

Competitive equilibrium prices are useful because they ensure a certain stability in the bids. If agents act purely as price takers, then they are satisfied with the given allocation, because it maximizes their utility. The auctioneer is also satisfied because no impression that could have generated more revenue goes unallocated.

Of course, agents may realize that they are not in fact price-takers, and that the prices—being dual variables of the allocation LP—should vary as the bid trees are changed, either in structure or value. It is therefore instructive to consider what incentives the agents may have to alter their bid trees. Ideally, we would like to reach a scenario where the agents are satisfied with their current bid trees as far as the allocation and prices that result. In this kind of equilibrium situation, the bid trees remain static, which lessens the burden on the system. The next section addresses this design issue.

7.6 Incentives

To achieve an *ex post* Nash equilibrium, we should implement Vickrey payments. Unfortunately, it may not be possible to achieve Vickrey payments in a competitive equilibrium, as the following example shows.

Example. There are two bidders $\{a, b\}$ and two identical impressions available. The

valuations are:

$$a : \textcircled{2} \qquad b : \textcircled{3}\ 1$$

The efficient allocation gives 1 impression to each, for a total value of 5. The only possible CE price is $p = 2$: any $p < 2$ makes agent a desire 2 units, whereas any $p > 2$ makes it desire 0 units. But under the VCG mechanism, agent a makes a surplus of 2 because this is its marginal contribution to efficiency, and hence its Vickrey payment of must be 0.

In this model, Vickrey payments cannot be priced because the price space is not rich enough—we only consider order 1 prices. We saw in Section 3.5 that with substitutes valuations, only order 3 prices can always price the Vickrey payoff point. One possibility would be to have a concise representation of such order 3 prices. For instance, we could simply quote the bid trees back to the bidders together with a discount, as in the elicitation scheme of Chapter 5.

This does not seem appropriate for this domain, because bidders can enter or leave the system at any time in a typical online ad auction, and so it is necessary to provide informative prices to new arrivals. Order 1 prices are very informative: they are easy to interpret, and apply to everyone. Order 3 prices, on the other hand, mean that there is no useful information for new entrants: they have to develop a bid tree and submit it to see if they can afford anything at all. Order 1 prices also have the advantage that they can be easily integrated into optimization problems on the bidder side—for instance, the problem of answering a demand query—especially if the optimizations rely on linear programming methods.

An alternative then is to quote the order 1 prices that minimize the incentives for

bidders to switch positions. This may be a good compromise if bidders are bounded-rational and would not notice or bother to switch when this would yield just a small improvement in payoff. In this case, Day and Milgrom [30] have shown that the CE prices that minimize the seller's revenues (and hence maximize the bidders' surplus) maximize the incentives for truthful reporting (see also Parkes et al. [84] for a related result in the context of combinatorial exchanges). Therefore, implementing the smallest order 1 CE price vector $\underline{p}$ leads to the most stable system, if we restrict ourselves to linear prices.

7.7 Bidder Feedback

Expressive bidding is appealing to advertisers because they can precisely specify their values for different kinds of impressions. On the other hand, it may prove daunting for certain advertisers to figure out how to update their bid trees to obtain more volume. In this section, we describe a simple bidder feedback mechanism that can suggest bid tree updates to bidders who want to increase their volume for certain kinds of impressions.

Suppose bidder i wants to receive at least d_{iT} impressions from sites in $T \subseteq M$, where $T \in \mathcal{T}_i$ is a node in bidder i's bid tree. For example, the advertiser with the bid tree in Figure 7.1(b) may only receive 10,000 impressions from FL, and want to increase FL impressions to 20,000. Naturally, the advertiser should first ensure that the volume constraint for the node is $c_{iT} \geq d_{iT}$. If this change still does not give the desired volume, the bid b_{iT} should be increased.

Suppose we introduce the constraint

$$\sum_{j \in T} x_i(j) \geq d_{iT} \tag{7.16}$$

into the linear program given in Section 7.4 to find an efficient allocation. Let λ be the dual variable corresponding to this constraint. The following lemma confirms that λ is informative feedback to the bidder.

Lemma 15 *Suppose bidder i increases b_{iT} to $b_{iT} + \lambda$ in its bid tree, and leaves the bids in the other nodes unchanged. Then there exists an efficient allocation, with respect to the new profile of bid trees, in which i receives at least d_{iT} impressions from $T \subseteq M$.*

Proof. Consider the primal program with constraint (7.16) added. By strong duality, a solution x is primal optimal for this program if and only if it is feasible and optimal for the program obtained by dualizing constraint (7.16). This latter program has the objective

$$\max \sum_{i \in N} \sum_{S \in \mathcal{T}_i} \sum_{j \in T} b'_{iS} x_i(j) - \lambda d_{iT}$$

where

$$b'_{iS} = \begin{cases} b_{iS} + \lambda & \text{if } S = T \\ b_{iS} & \text{otherwise} \end{cases}$$

We can drop the trailing constant λd_{iT} from the objective, which recovers the original allocation LP, except that the bid vector b has been replaced with the updated bid vector b'. Hence x is an optimal solution when bidder i increases b_{iT} by λ, and since it was feasible for the original program, it satisfies (7.16). □

The lemma shows that, just as bids can be specified at various levels of granularity, local updates to the bid tree can affect volumes at different levels of granularity. For

instance, if a bidder wishes to increase overall impressions regardless of origin, the feedback scheme would suggest an appropriate bid increase at the root node.

The feedback mechanism only advises on how to update values within a bid tree, not on how to update the tree's structure. One way to elicit structure, in line with the approach of the preference elicitation framework, would be to develop a learning algorithm for bid trees using value and demand queries. This is an important next step for future work.

Chapter 8

Conclusions

The combinatorial allocation problem abstracts the resource allocation problems of several domains in both the private and public sector, such as bandwidth allocation, vehicle routing, spectrum allocation, and industrial procurement. Internet advertising is also a domain where complicated preferences increasingly need to be modeled, as advertisers demand more control over the targeting of their online campaigns. This dissertation described, instantiated, and implemented a preference elicitation framework for the CAP that in principle allows for a variety of representations for agent preferences.

The central idea is to embed learning algorithms for the various representations into the process of resource allocation. The approach is not to simply learn the valuations, then solve the winner determination problem. An instantiation of the framework will incrementally recover the agents' preferences, and periodically compute an allocation together with tentative competitive equilibrium prices to check whether enough value information has been elicited to solve the allocation problem.

In general, winner determination is solved each time the underlying learning algorithms need to issue equivalence or demand queries.

In the framework, prices can equal manifest valuations, but the two can also be decoupled. We can present discounted valuations as prices, or use a different pricing engine entirely. This decoupling is similar in spirit to auctions such as RAD [56] that compute approximate linear CE prices at each round to provide feedback, while the agents provide package bids. The framework allows for nonlinear prices, which motivates the notion of a *pricing language* to succinctly represent prices, in parallel to the notion of a bidding language. Other package auctions that use nonlinear prices include *i*Bundle [78] and dVSV [31], but these are restricted to the XOR language to represent prices, because they equate prices with bids. In the proposed framework, preference elicitation corresponds to a "bid update," and the prices then get updated accordingly when the pricing engine recomputes them. This allows for flexibility in the choice of representations.

In the introduction we saw that iterative auctions have many advantages over their single-shot counterparts, including the potential for improved efficiency, transparency, and privacy. However, adoption of iterative combinatorial auctions has been slow in industry, in part because current designs do not provide a convenient language for representing valuations. The preference elicitation framework proposed in this dissertation is an early step towards the development of iterative package auctions that could allow for representations tailored to the relevant application domains.

8.1 Review

Multi-agent preference elicitation lies at the interface of artificial intelligence, operations research, and microeconomics. In the process of instantiating the framework, this dissertation provided contributions that fall within each of these areas and that are interesting in their own right.

Chapter 2 presented background results on the VCG mechanism and the core. VCG payments align the agents' incentives with the system-wide objective, ensuring that agents truthfully respond to queries about their valuations in equilibrium (dominant-strategy equilibrium for single-shot auctions, *ex post* Nash equilibrium for iterative auctions). This fact motivated an extension to the elicitation framework for computing VCG payments in Chapter 5. The core captures the feasible gains from trade that could plausibly occur if the agents negotiated an allocation amongst themselves. In the context of the preference elicitation framework, the concept arises because competitive equilibrium prices can be derived from the agents' valuations and core payoffs.

Chapter 3 presented results on competitive equilibrium with indivisibilities when agents have superadditive valuations, a common regime in applications of package auctions. The three main contributions related superadditive valuations and order 2 pricing: (1) I gave a constructive, non-algorithmic proof that anonymous, nonlinear competitive equilibrium prices exist when bidders have superadditive valuations—this fact was first shown by Parkes [79], whose proof relied on the properties of the *i*Bundle(d) auction; (2) I strengthened the latter result to show that with superadditive valuations, anonymous, nonlinear competitive equilibrium prices price the core;

and (3) I established that the class of superadditive valuations is a maximal class containing the single-minded valuations that ensures anonymous, nonlinear competitive equilibrium prices exist. These results fill a gap in our understanding of competitive equilibrium with indivisibilities; analogous results already existed for order 1 and 3 pricing.

Chapter 3 also gives results on the communication requirements of computing the VCG outcome (an efficient allocation together with Vickrey payments). It showed that the nondeterministic communication complexity of computing the VCG outcome was equal to the nondeterminstic communication complexity of computing UCE prices: any protocol that computes the VCG outcome also (implicitly) computes UCE prices. This parallels Nisan and Segal's [76] result that equates the communication complexities of computing an efficient allocation and a competitive equilbrium. I also showed how UCE prices can be constructed from CE prices for the main and marginal economies, and gave an example to show that computing Vickrey payments on top of the efficient allocation can strictly increase communication requirements.

In Chapter 4 we surveyed learning algorithms for the XOR, OR, and Polynomials languages that use value, demand, and equivalence queries; the algorithms for XOR and OR were novel. Chapter 4 also introduced the notion of a *pricing language*. Pricing languages are relevant in preference elicitation schemes for the CAP, and XOR is the only pricing language that has been used to date besides linear prices. I proposed Pseudo-additive representations as a novel pricing language, and made use of it in the iterative auction of Chapter 6.

Chapter 5 described the preference elicitation framework. If the underlying learn-

ing algorithms are polynomial-time, the elicitation process has polynomial communication complexity. These guarantees are formulated in terms of the number of items and agents, and also in terms of the minimal size of the agents' valuations in the chosen representations; this latter parameter allows for relevant complexity guarantees that focus the design problem on the choice of representation.

An empirical evaluation showed that Polynomial representations complement XOR representations, in the sense that the resulting elicitation process performs well with Polynomials when it performs poorly with XOR, and vice-versa, on standard benchmark distributions. This is convenient because XOR has been the only expressive language used in iterative auctions in the academic literature to date. These findings were all the more significant because the underlying agent valuations were represented not with Polynomials or XOR but with the OR language, and they agreed with basic intuition into which representations would perform better under different distribution settings. The scaling properties of the elicitation framework in terms of total queries and runtime were better than the worst-case guarantees suggested, when using the language of Polynomials while underlying valuations had succinct Polynomial representations, and similarly for XOR.

Chapter 6 presented an iterative auction that begins with linear prices, and introduces nonlinearities into the price space as needed to ensure a competitive equilibrium exists. The main insight was that the notion of a *pattern* can allow for LP formulations of the combinatorial allocation problem that are intermediate between Bikhchandani and Ostroy's [13] order 1 and 2 formulations. Adding bundles to a pattern leads to successively stronger formulations, and can be interpreted as adding cutting planes.

The iterative auction only uses demand queries, but there is also a variant that issues value queries to elicit value information over fewer rounds.

The latter variant can be used within the preference elicitation framework as a pricing engine, because responses to value queries are readily available from the proxies. This could be a favorable alternative to the pricing method used in the implementation described in Chapter 5. Recall that the implementation provides discounted valuations back as prices, which could be undesirable from privacy perspective: if prices are made public in the interest of transparency, then the bidders would all have succinct representations of each others' valuations, up to additive constants. This could be problematic in certain domains. If the bidders were firms in a procurement auction, for example, the structure of the price representations might reveal sensitive information about the structures of the firms' business plans. The iterative auction of Chapter 6 computes a pseudo-additive representation of prices, which may differ from the representations used by the proxies.

Chapter 7 applied the principles of expressive auction design to the domain of Internet advertising. We first surveyed the properties of current position auctions for sponsored search. I then proposed a bidding language for advertisers to express their valuations over sets of impressions for display advertising. The bid tree bidding language is convenient because it allows advertisers to specify values according to different attributes of the impresssions, at varying levels of granularity, and also to specify volume constraints to control exposure. The language was developed with the insights of Chapter 3 in mind: bid trees represent substitutes valuations, which implies that order 1 CE prices exist, as does a concise LP formulation of the alloca-

tion problem (whose dual gives prices). Linear pricing is appropriate for this domain because online advertisers range from multi-national corporations to small local businesses, so simple, informative price feedback is crucial. The constant arrival and exit of new advertisers in an online advertising system also means that allocations have to be recomputed very quickly, ideally close to real time.

8.2 Future Work

There are several immediate directions for future work. One outstanding problem is to provide a better lower bound on the communication requirements of the VCG mechanism. The lower bound does not yet match the upper bound of an n-fold increase in communication. I suspect that a fooling set involving substitutes valuations could give a tight lower bound.

The framework was only evaluated empirically on myopic best-response agents. It would be informative to test the extended version that converges to UCE prices rather than just CE prices, to get a sense of the added communication that VCG payments entail. It would also be informative to test the effect of anonymous prices on the number of queries, rounds, and the runtime. If the underlying representations are OR (which is the case with several benchmark distributions), valuations are superadditive and there exist anonymous prices. In this case, we could use the anonymous price construction of Chapter 3 to compute CE prices in the framework.

A broader research program is to develop a wider variety of bidding and pricing languages together with learning algorithms for use in the framework. For instance, a learning algorithm for the Pseudo-additive language is still lacking. Kernel meth-

ods [102] could also find application within the framework: given a set of bids, we could run a support vector machine with various kernels to obtain a concise representation of an agent's valuation.

Another line of investigation involves the choice of pricing language, and its effect on query complexity and the speed of elicitation. The naïve approach of quoting discounted valuations as prices works when there are few bidders, but as the number of bidders grows so does the number of losing bidders. The naïve approach will learn the valuations of these losing bidders entirely. With large numbers of bidders, however, core convergence results state that linear prices should almost clear the market, so approaches such as the cutting-plane auction of Chapter 6 should perform well. This intuition remains to be validated empirically.

Bibliography

[1] Gagan Aggarwal, Ashish Goel, and Rajeev Motwani. Truthful auctions for pricing search keywords. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, Ann Arbor, MI, 2006.

[2] Arne Andersson, Mattias Tenhunen, and Fredrik Ygge. Integer programming for combinatorial auction winner determination. In *Proceedings of the 4th International Conference on Multiagent Systems*, Boston, MA, 2000.

[3] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[4] Dana Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1987.

[5] Lawrence M. Ausubel. An efficient dynamic auction for heterogeneous commodities. *American Economic Review*, 2006. Forthcoming.

[6] Lawrence M. Ausubel, Peter Cramton, R. Preston McAfee, and John McMillan. Synergies in wireless telephony: Evidence from the broadband PCS auction. *Journal of Economics and Management Strategy*, 6(3):497–527, 1997.

[7] Lawrence M Ausubel and Paul R Milgrom. Ascending auctions with package bidding. *Frontiers of Theoretical Economics*, 1:1–42, 2002.

[8] Dimitri P. Bertsekas. The auction algorithm: a distributed relaxation method for the assignment problem. *Annals of Operations Research*, 14:105–123, 1988.

[9] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, Belmont, MA, 1997.

[10] Martin Bichler, Andrew Davenport, Gail Hohner, and Jayant Kalagnanam. *Industrial Procurement Auctions*, chapter 23, pages 593–612. MIT Press, 2006.

[11] Sushil Bikhchandani, Sven de Vries, James Schummer, and Rakesh V. Vohra. Linear programming and Vickrey auctions. In Brenda Dietrich and Rakesh Vohra, editors, *Mathematics of the Internet: E-Auction and Markets*, pages

75–116. IMA Volumes in Mathematics and its Applications, Springer-Verlag, 2001.

[12] Sushil Bikhchandani and John W. Mamer. Competitive equilibrium in an exchange economy with indivisibilities. *Journal of Economic Theory*, 74:385–413, 1997.

[13] Sushil Bikhchandani and Joseph M. Ostroy. The package assignment model. *Journal of Economic Theory*, 107:377–406, 2002.

[14] Robert G. Bland. New finite pivoting rules for the simplex method. *Mathematics of Operations Research*, 2:103–107, 1977.

[15] Avrim Blum, Jeffrey Jackson, Tuomas Sandholm, and Martin Zinkevich. Preference elicitation and query learning. In *Proceedings of the 16th Annual Conference on Computational Learning Theory*, Washington, DC, 2003.

[16] Tilman Börgers, Ingemar Cox, Martin Pesendorfer, and Vaclav Petricek. Equilibrium bids in auctions of sponsored links: Theory and evidence. Working paper, November 2006.

[17] Craig Boutilier. Solving concisely expressed combinatorial auction problems. In *Proceedings of the 19th National Conference on Artificial Intelligence*, pages 359–366, 2002.

[18] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. CP-nets: Tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 2003.

[19] Craig Boutilier, Ronen I. Brafman, Christopher W. Geib, and David Poole. A constraint-based approach to preference elicitation and decision making. In *AAAI Spring Symposium on Qualitative Decision Theory*, pages 19–28, Palo Alto, CA, 1997.

[20] Craig Boutilier and Holger H. Hoos. Bidding languages for combinatorial auctions. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1211–1217, 2001.

[21] Robin D. Burke, Kristian J. Hammond, and Edwin Cooper. Knowledge-based navigation of complex information spaces. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 462–468, 1996.

[22] Ruggiero Cavallo, David C. Parkes, Adam Juda, Adam Kirsch, Alex Kulesza, Sébastien Lahaie, Benjamin Lubin, Loizos Michael, and Jeffrey Shneidman.

TBBL: A tree-based bidding language for iterative combinatorial exchanges. In *IJCAI Multidisciplinary Workshop on Advances in Preference Handling*, 2005.

[23] Li Chen and Pearl Pu. Survey of preference elicitation methods. Technical report, Ecole Polytechnique Fédérale de Lausanne, 2004.

[24] Vašek Chvátal. *Linear Programming*. W. H. Freeman, New York, NY, 1983.

[25] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[26] Vincent Conitzer and Tuomas Sandholm. Computational criticisms of the revelation principle. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, New York, NY, 2004.

[27] Peter Cramton. Ascending auctions. *European Economic Review*, 42(3–5):745–756, 1998.

[28] Peter Cramton, Yoav Shoham, and Richard Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006.

[29] Vladimir Danilov, Gleb Koshevoy, and Kazuo Murota. Discrete convexity and equilibria in economies with indivisible goods and money. *Mathematical Social Sciences*, 41(3):251–273, 2001.

[30] Robert Day and Paul Milgrom. Core-selecting auctions. *International Journal of Game Theory*, 2007. Forthcoming.

[31] Sven de Vries, James Schummer, and Rakesh V. Vohra. On ascending Vickrey auctions for heterogeneous objects. *Journal of Economic Theory*, 2005. Forthcoming.

[32] Sven de Vries and Rakesh V. Vohra. Combinatorial auctions: A survey. *Informs Journal on Computing*, 2002. Forthcoming.

[33] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the Generalized Second Price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 2005. Forthcoming.

[34] Jack Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1:127–136, 1971.

[35] Márta Eső. *Parallel Branch and Cut for Set Partitioning*. PhD thesis, School of Operations Research and Industrial Engineering, Cornell University, 1999.

[36] Ronald Fadel and Ilya Segal. The communication cost of selfishness. *Journal of Economic Theory*, 2007. Forthcoming.

[37] Juan Feng, Hemant K. Bhargava, and David M. Pennock. Implementing sponsored search in web search engines: Computational evaluation of alternative mechanisms. *INFORMS Journal on Computing*, 2005. Forthcoming.

[38] Donald F. Ferguson, Christos Nikolaou, Jakka Sairamesh, and Yechiam Yemini. *Economic models for allocating resources in computer systems*, chapter 7, pages 156–183. World Scientific Publishing Company, 1996.

[39] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 548–553, Stockholm, Sweden, 1999.

[40] Stephen Glaiser and Michael Beesley. Bidding for tendered bus routes in London. *Transportation Planning and Technology*, 15:349–366, 1991.

[41] Jerry Green and Jean-Jacques Laffont. Characterization of satisfactory mechansims for the revelation of preferences for public goods. *Econometrica*, 45:427–438, 1977.

[42] Theodore Groves. Efficient collective choice when compensation is possible. *Review of Economic Studies*, 46:227–241, 1979.

[43] Faruk Gul and Ennio Stacchetti. Walrasian equilibrium with gross substitutes. *Journal of Economic Theory*, 87:95–124, 1999.

[44] Godfrey H. Hardy, John E. Littlewood, and George Pólya. *Inequalities*. Cambridge University Press, 1934.

[45] Gail Hohner, John Rich, Ed Ng, Grant Reid, Andrew Davenport, Jayant Kalagnanam, Ho Soo Lee, and Chae An. Combinatorial and quantity discount procurement auctions with mutual benefit at Mars, inc. *Interfaces*, 33:23–35, 2003.

[46] Bengt Holmstrom. Groves schemes on restricted domains. *Econometrica*, 47(5):1137–1144, 1979.

[47] Garud Iyengar and Anuj Kumar. Characterizing optimal keyword auctions. In *Proceedings of the 2nd Workshop on Sponsored Search Auctions*, Ann Arbor, MI, 2006.

[48] Ralph Cassady Jr. *Auctions and Auctioneering*. University of Califrnia Press, Berkeley, CA, 1967.

[49] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[50] Alexander S. Kelso and Vincent P. Crawford. Job matching, coalition formation, and gross substitutes. *Econometrica*, 50:1483–1504, 1982.

[51] Paul Klemperer. Almost common value auctions: the 'wallet game' and its applications to takeover battles and airwaves auctions. *European Economic Review*, 42, 1998.

[52] Tjalling C. Koopmans and Martin Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25(1):53–76, January 1957.

[53] Bernhard Korte and Jens Vygen. *Combinatorial Optimization*. Springer, 2002.

[54] Vijay Krishna. *Auction Theory*. Academic Press, 2002.

[55] Vijay Krishna and Motty Perry. Efficient mechanism design. Technical report, Pennsylvania State University, 2000.

[56] Anthony M. Kwasnica, John O. Ledyard, David Porter, and Christine DeMartini. A new and improved design for multiobject iterative auctions. *Management Science*, 51:419–434, 2005.

[57] Sébastien Lahaie. An analysis of alternative slot auction designs for sponsored search. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, Ann Arbor, MI, 2006.

[58] Sébastien Lahaie, Florin Constantin, and David C. Parkes. More on the power of demand queries in combinatorial auctions: Learning atomic languages and handling incentives. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, 2005.

[59] Sébastien Lahaie and David C. Parkes. Applying learning algorithms to preference elicitation. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 180–188, New York, NY, 2004. ACM Press.

[60] Sébastien Lahaie and David M. Pennock. Revenue analysis of a family of ranking rules for keyword auctions. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, San Diego, CA, 2007.

[61] John O. Ledyard, Mark Olson, David Porter, Joseph A. Swanson, and David P. Torma. The first use of a combined value auction for transportation services. Technical report, California Institute of Technology, 2000.

[62] Herman B. Leonard. Elicitation of honest preferences for the assignment of individuals to positions. *The Journal of Political Economy*, 91(3):461–479, 1983.

[63] Kevin Leyton-Brown, Eugene Nudelman, Galen Andrew, Jim McFadden, and Yoav Shoham. Learning the empirical hardness of optimization problems: the case of combinatorial auctions. In *CP*, pages 556–572, 2003.

[64] Kevin Leyton-Brown, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proceedings of the second ACM Conference on Electronic Commerce*, pages 66–76, Minneapolis, MN, 2000.

[65] Greg Linden, Steve Hanks, and Neal Lesh. Interactive assessment of user preference models: the automated travel assistant. In *Proceedings of User Modeling*, 1997.

[66] David G. Luenberger. *Linear and Nonlinear Programming*. Wiley, New York, NY, 1984.

[67] Andreu Mas-Colell, Michael D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[68] John McMillan. Selling spectrum rights. *Journal of Economic Perspectives*, 8:145–162, 94.

[69] Paul Milgrom. *Putting Auction Theory to Work*. Cambridge University Press, 2004.

[70] Paul Milgrom and Robert J. Weber. A theory of auctions and competitive bidding. *Econometrica*, 50:1089–1122, 1982.

[71] Debasis Mishra and David C. Parkes. Ascending price Vickrey auctions for general valuations. *Journal of Economic Theory*, 2005. Forthcoming.

[72] Kazuo Murota. *Discrete Convex Analysis*. SIAM, 2003.

[73] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.

[74] Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proceedings of the second ACM Conference on Electronic Commerce*, pages 1–12, Minneapolis, MN, 2000.

[75] Noam Nisan and Ilya Segal. The communication requirements of efficient allocations and supporting Lindahl prices. Working Paper, Hebrew University, 2003.

[76] Noam Nisan and Ilya Segal. The communication requirements of efficient allocations and supporting prices. *Journal of Economic Theory*, 2006. Forthcoming.

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

[78] David C. Parkes. *i*Bundle: An efficient ascending price bundle auction. In *Proceedings of the first ACM Conference on Electronic Commerce*, pages 148–157, Denver, CO, 1999.

[79] David C. Parkes. *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, May 2001.

[80] David C. Parkes. Notes on indirect and direct implementations of core outcomes in combinatorial auctions. Technical report, Harvard University, 2002.

[81] David C. Parkes. Price-based information certificates for minimal-revelation combinatorial auctions. In Padget et al., editor, *Agent-Mediated Electronic Commerce IV, LNAI 2531*, pages 103–122. Springer, 2002.

[82] David C. Parkes. Auction design with costly preference elicitation. In *Special Issues of Annals of Mathematics and AI on the Foundations of Electronic Commerce*, 2003. Forthcoming.

[83] David C. Parkes, Ruggiero Cavallo, Nick Elprin, Adam Juda, Sébastien Lahaie, Benjamin Lubin, Loizos Michael, Jeffrey Shneidman, and Hassan Sultan. ICE: An iterative combinatorial exchange. In *Proceedings of the 6th ACM Conference on Electronic Commerce*, pages 249–258, Vancouver, Canada, 2005. ACM Press.

[84] David C. Parkes, Jayant Kalagnanam, and Márta Eső. Achieving budget-balance with Vickrey-based payment schemes in exchanges. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 1161–1168, Seattle, WA, 2001.

[85] David C. Parkes, Michael O. Rabin, Stuart M. Shieber, and Christopher Thorpe. Practical secrecy-preserving, verifiably correct and trustworthy auctions. Technical report, Harvard University, 2007.

[86] David C. Parkes and Lyle H. Ungar. An ascending-price generalized Vickrey auction. Technical report, Harvard University, 2002.

[87] David Porter, Stephen Rassenti, Anil Roopnarine, and Vernon Smith. Combinatorial auction design. *Proceedings of the National Academy of Sciences*, 100:11153–11157, 2003.

[88] Stephen J. Rassenti, Vernon L. Smith, and Robert L. Bulfin. A combinatorial mechanism for airport time slot allocation. *Bell Journal of Economics*, 13:402–417, 1982.

[89] Michael H. Rothkopf, Aleksandar Pekeč, and Ronald M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[90] Michael H. Rothkopf, Thomas J. Teisberg, and Edward P. Kahn. Why are Vickrey auctions rare? *Journal of Political Economy*, 98:94–109, 1990.

[91] Stuart Russell and Peter Norvig. *Artificial Intelligence: a Modern Approach*. Prentice Hall, 2002.

[92] Yuko Sakurai, Makoto Yokoo, and Shigeo Matsuraba. A limitation of the generalized Vickrey auction in electronic commerce: Robustness against false-name bids. In *Proceedings of the 19th National Conference on Artificial Intelligence*, San Jose, CA, 2004.

[93] Tuomas Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 1999.

[94] Tuomas Sandholm. Expressive commerce and its application to sourcing: How we conducted $35 billion of generalized combinatorial auctions. In *Proceedings of the Conference on Innovative Applications of Artificial Intelligence*, Vancouver, Canada, 2007.

[95] Tuomas Sandholm and Subhash Suri. BOB: Improved winner determination in combinatorial auctions and generalizations. *Artificial Intelligence*, 145:33–58, 2003.

[96] Tuomas Sandholm and Subhash Suri. Side constraints and non-price attributes in markets. *Games and Economic Behavior*, 55:321–330, 2006.

[97] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. CABOB: A fast optimal algorithm for combinatorial auctions. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 1102–1108, Seattle, WA, 2001.

[98] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. Winner determination in combinatorial auction generalizations. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems*, pages 69–76, Bologna, Italy, 2002.

[99] Robert Schapire and Linda Sellie. Learning sparse multivariate polynomials over a field with queries and counterexamples. *Journal of Computer and Systems Sciences*, 52(2):201–213, 1996.

[100] Alexander Schrijver. *Combinatorial Optimization*. SIAM, 2003.

[101] Lloyd S. Shapley and Martin Shubik. The assignment game I: The core. *International Journal of Game Theory*, 1:111–130, 1972.

[102] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[103] Hideo Shimazu. Expertclerk: Navigating shoppers' buying process with the combination of asking and proposing. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, Seattle, WA, 2001.

[104] Leslie Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[105] Hal R. Varian. Position auctions. *International Journal of Industrial Organization*, 2006. Forthcoming.

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

[107] Rakesh V. Vohra. *Advanced Mathematical Economics*. Routledge, 2005.

[108] William E. Walsh, Michael P. Wellman, and Fredrik Ygge. Combinatorial auctions for supply-chain formation. In *Proceedings of the second ACM Conference on Electronic Commerce*, pages 260–269, Minneapolis, MN, 2000.

[109] Laurence A. Wolsey. *Integer Programming*. John Wiley & Sons, 1998.

[110] Peter R. Wurman and Michael P. Wellman. A$k$ba: A progressive, anonymous-price combinatorial auction. In *Proceedings of the second ACM Conference on Electronic Commerce*, pages 21–29, Minneapolis, MN, 2000.

[111] Martin Zinkevich, Avrim Blum, and Tuomas Sandholm. On polynomial-time preference elicitation with value-queries. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, San Diego, CA, June 2003.