ITERATIVE COMBINATORIAL AUCTIONS: ACHIEVING ECONOMIC AND COMPUTATIONAL EFFICIENCY

David Christopher Parkes

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2001

Lyle H. Ungar
Supervisor of Dissertation

Val Tannen
Graduate Group Chair

Acknowledgements

None of this would be possible without my advisor, Lyle Ungar. You cannot imagine how grateful I am for your constant support and guidance. You are an amazing inspiration. I hope this dissertation (and the tables!) will go some way towards expressing my thanks. I am also deeply indebted to Patrick Harker for helping to start me off in the right direction, and to my other committee members, Mitch Marcus, Sampath Kannan, and especially to Michael Wellman for insightful feedback and being a great mentor. I must also thank Prof. Bird at Oxford, who first got me excited about computer science. Thanks also to Lloyd for the animal crackers and bounded-rationality, and to Marta and Jayant for being great researcher friends last summer and for the white board proofs. Other people that have helped along the way include Peter Wurman, Amy Greenwald, Tuomas Sandholm, William Walsh, Fredrik Ygge, Vijay Chandru, Bernardo Huberman, Rakesh Vohra, and Sushil Bikchandani. Matthew, Doug, Harold, Rinaldo, Jon, Kathy and Panos, my PhD-seeking friends. Mike Felker. Penn cycling, 1224 Rodman, Randy, Ay-Ay-Ay, Cat W.,... just an amazing few years. My friends in Philadelphia. I am going to miss you all so much. Especially Rob. Moooo. Dan, I know how proud you are. Thanks for believing in me all the way through. Laura, Mum and Dad. I miss you and love you. Thanks for keeping so close despite being so far away. Finally Phil. This book of mathematical dreams is all for my star dreamer. I love you.

Abstract

ITERATIVE COMBINATORIAL AUCTIONS:
ACHIEVING ECONOMIC AND COMPUTATIONAL EFFICIENCY
David Christopher Parkes
Supervisor: Lyle H. Ungar

A fundamental problem in building open distributed systems is to design mechanisms that compute optimal system-wide solutions despite the self-interest of individual users and computational agents. Classic game-theoretic solutions are often prohibitively expensive computationally. For example, the Generalized Vickrey Auction (GVA) is an efficient and strategy-proof solution to the combinatorial allocation problem (CAP), in which agents demand bundles of items, but every agent must reveal its value for all possible bundles and the auctioneer must solve a sequence of NP-hard optimization problems to compute the outcome.

I propose *i*Bundle, an *iterative* combinatorial auction in which agents can bid for combinations of items and adjust their bids in response to bids from other agents. *i*Bundle computes the efficient allocation in the CAP when agents follow myopic best-response bidding strategies, bidding for the bundle(s) that maximize their surplus taking the current prices as fixed. *i*Bundle solves problems without complete information revelation from agents and terminates in competitive equilibrium. Moreover, an agent can follow a myopic best-response strategy with approximate values on bundles, for example with lower- and upper- bounds.

My approach to iterative mechanism design decomposes the problem into two parts. First, I use linear programming theory to develop an efficient iterative auction under the assumption that agents will follow a myopic best-response bidding strategy. Second, I extend the approach to also compute Vickrey payments at the end of the auction. This

makes myopic best-response a sequentially-rational strategy for agents in equilibrium, inheriting many of the useful game-theoretic properties of the GVA.

*i*Bundle implements a primal-dual algorithm, CombAuction, for the CAP, computing a feasible primal (the provisional allocation) and a feasible dual (the ask prices) that satisfy complementary slackness conditions. An extended auction, *i*Bundle Extend&Adjust, interprets a primal-dual algorithm, VickAuction, as an iterative auction. VickAuction computes the efficient allocation and Vickrey payments with only best-response information from agents. Experimental results demonstrate that *i*Bundle Extend&Adjust, which keeps *i*Bundle open for a second phase before adjusting prices towards Vickrey payments, computes Vickrey payments across a suite of problems.

Contents

List of Tables

List of Figures

Preface

This is a slight revision of my dissertation, as handed in on May 4, 2001 to College Hall. This version is available as Technical report MS-CIS-01-17, Department of Computer and Information Science, University of Pennsylvania.

David C. Parkes, May 24, 2001.

Chapter 1

Introduction

The Internet embodies a new paradigm of distributed open computer networks, in which agents— users and computational devices —are not cooperative, but self-interested, with private information and goals. A fundamental problem in building these systems is to design *incentive-compatible* protocols, which compute optimal system-wide solutions despite the self-interest of individual agents. This emerging area of study, called *computational mechanism design*, is at the interface of game theory, artificial intelligence, and algorithmic theory.

Examples of the many interesting applications of mechanism design in open systems include: (a) network routing problems, with self-interested packets and routers; (b) procurement problems in electronic commerce between businesses and suppliers; (c) logistics problems, with task allocation across multiple self-interested shipping companies; (d) scheduling problems, for example to schedule time slots at airport gates across self-interested airlines.

We can view these problems as distributed optimization problems, with an objective function that depends on the *private information* of the agents in the system. A rational self-interested agent will choose to reveal incomplete, and perhaps untruthful, information about its goals and preferences if that leads to an individually preferable outcome. The central goal in mechanism design is to address this problem of agent self-interest, and design incentives to encourage agent behavior that leads to good system-wide solutions to distributed multi-agent optimization problems. One classic approach is to design incentives for agents to provide *truthful* information about their preferences over different outcomes, and compute an optimal system-wide solution with this information.

Given that market-based mechanisms have proved able to coordinate the activities of

many autonomous individuals in human societies, it is perhaps natural to look to economic principles to design coordination mechanisms in computational systems. Indeed, the explosion of Internet-based commerce creates a huge demand for efficient market-based mechanisms, for example to support automated negotiation across and within groups of individuals and businesses. Market-based mechanisms are a very natural way to respect the autonomy and information decentralization in open systems, and promise to revolutionize how we design and evaluate open computer systems. In particular, *auction-based* mechanisms can often provide enough structure to enable strong theoretical claims about the strategies that agents will select and the optimality properties of final solutions.

Exploring the interface between economic mechanisms and distributed agent-based optimization problems exposes a number of deep computational problems. Limited and/or costly computation, both at the network and at the level of distributed computational agents, coupled with the inherent combinatorial complexity of many interesting problem domains (e.g. those in scheduling and resource allocation) can quickly break naive implementations of classic game-theoretic mechanisms. Yet, computation and self-interest interact in non-obvious ways: while approximate solutions can destroy the incentive properties of a mechanism, agent bounded-rationality can also be used to design mechanisms that cannot be manipulated without solving an intractable problem.

1.1 Computational Mechanism Design

The challenge in computational mechanism design is to resolve the tensions between what one might choose to do game-theoretically and what is desirable computationally. In some cases the best game-theoretic solution also provides useful computational benefits. The most obvious example is the concept of *dominant strategy implementation*, which implies that each agent has an optimal strategy irrespective of the preferences or strategies of other agents. This is useful computationally because agents do not need to model or deliberate about the strategies of the other agents in the system. We are not always so lucky. To give a counterexample, a *direct-revelation* dominant strategy implementation, in which every agent must compute and reveal its complete preferences over all possible outcomes, is a useful simplifying concept game-theoretically but often intractable computationally.

A useful mechanism must control both the computational costs of the auctioneer (or

in general, of the mechanism infrastructure) *and* the computational costs of the agents, while retaining useful game-theoretic properties that handle agent self-interest.

In what follows I introduce the mechanism design problem, and consider computational problems with the classic game-theoretic approach, in particular in application to combinatorial problem domains. Then I briefly highlight some important computational costs in mechanism implementation, and introduce a number of different approaches to make mechanisms more computationally reasonable.

1.1.1 The Mechanism Design Problem

Consider a system with $\mathcal{I}$ agents, indexed $i = 1, \ldots, I$, and a set of outcomes $\mathcal{O}$. Each agent has private information about its *utility* for different outcomes, which is a quantitative measure of its "happiness" given each outcome. It is useful to think of an agent having a *type*, denoted $\theta_i \in \Theta_i$, that determines its utility over different outcomes. The set Θ_i represents all possible preferences available to agent i. We can write $u_i(o, \theta_i)$ to denote the utility of agent with type θ_i for outcome $o \in \mathcal{O}$. With this, then outcome o_1 is preferred to outcome o_2 by agent i, written $o_1 \succ o_2$, if and only if $u_i(o_1, \theta_i) > u_i(o_2, \theta_i)$.

The *implementation problem*, illustrated in Figure 1.1, is to compute the solution to a *social choice function*, $f : (\Theta_1, \times, \ldots, \times, \Theta_I) \rightarrow \mathcal{O}$, that selects an optimal outcome $o^* = f(\theta)$ based on the types $\theta = (\theta_1, \ldots, \theta_I)$ of all agents.

A common social choice function selects an outcome to maximize total utility over agents:

$$f(\theta) = \arg\max_{o \in \mathcal{O}} \sum_i u_i(o, \theta_i), \qquad \text{for all } \theta \in \Theta$$

This is the classic utilitarian objective, known as allocative-efficiency in allocation problems.

The *mechanism design* problem is to solve the implementation problem with self-interested agents that have private information about their preferences. Essentially a mechanism defines the "rules of a game"; i.e., the actions available to agents and the method that is used to compute the outcome based on those actions.

Auctions are simple mechanisms for resource allocation, in which the actions available to agents are to submit bids and the outcome is computed, for example, to maximize revenue given bids.

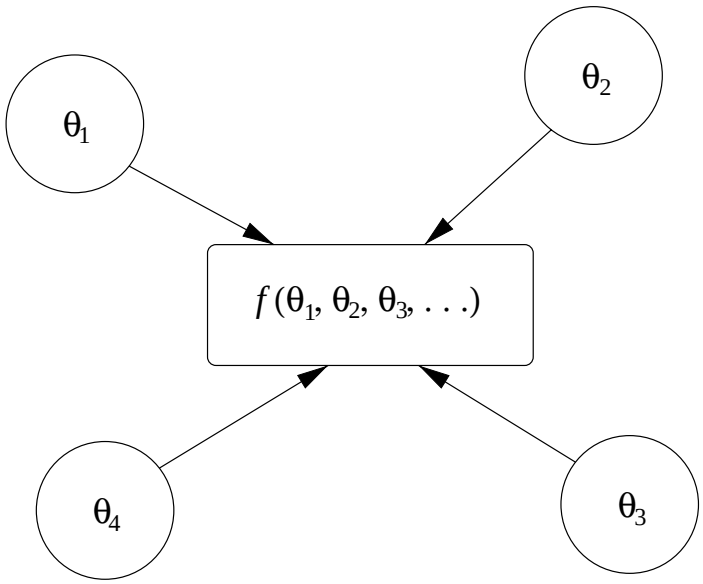


Figure 1.1: The implementation problem.

A game-theoretic approach is central to mechanism design. Game theory is a method to study a system of self-interested agents in conditions of strategic interaction, with rational agents modeled as expected-utility maximizers. Game-theoretic analysis computes the *equilibrium outcome* in a mechanism, for particular agent preferences, in which every agent plays an expected-utility maximizing best-response to every other agent. A number of solution concepts are defined, each of which makes different assumptions about the information available to agents and methods used by agents to select strategies.

With this, a mechanism is said to *implement* a particular social choice function, or solve an implementation problem, if the solution to the social choice function is computed by the mechanism in a game-theoretic equilibrium with self-interested agents.

1.1.2 The Classic Vickrey-Clarke-Groves Solution

One particularly useful solution concept in mechanism design is *dominant strategy* implementation, in which each agent has the same optimal strategy *whatever* the strategies and preferences of other agents. A mechanism that implements a desired social choice function in dominant strategy is a robust solution to an implementation problem, because it makes very few assumptions about the information available to agents, or about agents' beliefs

about the rationality of other agents. Informally, we might say that a dominant strategy solution "removes game theory" from the problem. Agents can participate without modeling the preferences or strategies of other agents.

This dominant-strategy solution concept is achieved in the important Vickrey-Clarke-Groves (VCG) family of mechanisms, in which an agent's dominant strategy is to *truthfully reveal* its preferences to the mechanism, whatever the strategies or preferences of other agents. This property of dominant-strategy truth-revelation is known as *strategy-proofness*. Moreover, in the context of resource allocation problems the VCG mechanism implements the *allocatively-efficient* solution, or the resource allocation that maximizes value over all agents. In fact, there is quite a strong sense in which *any* strategy-proof and efficient mechanism must compute the outcome of the VCG mechanism.

Unfortunately, the VCG mechanism provides an *extremely centralized* solution to the resource allocation problem. Every agent must provide *complete information* about its preferences to the mechanism. There are many problems, for example the combinatorial allocation problem, in which this is intractable for an agent. Agents often have difficult local valuation problems [PUF99, Mil00a]. Resolving this tension between computational concerns and game-theoretic concerns, and moving towards an iterative implementation of the VCG mechanism, is a central contribution of this dissertation.

1.1.3 Computational Considerations

Computation occurs at two different levels within a mechanism:

- For the mechanism infrastructure: How much computation is required to compute the outcome of the mechanism, for example given bids from agents?

- For agents:
  (a) (*strategic complexity*) Is *game-theoretic reasoning* required to follow an optimal strategy, or does an agent have a *dominant strategy*?
  (b) (*valuation complexity*) Must an agent compute evaluate its preferences for all possible outcomes to compute its optimal strategy?

Approaches to reduce the computational demands on the mechanism include: introducing approximations; identifying tractable special-cases; and distributing computation to the agents in the system. The challenge in each case is to relax computational demands

without losing useful game-theoretic properties. Perhaps one can identify a set of axioms that an approximation algorithm must satisfy to retain strategy-proofness, or restrict problem instances to a smaller set of tractable special-cases. Section 3.2.1 in Chapter 3 discusses some interesting approaches in more detail.

The *strategic complexity* of a mechanism is closely linked to its game-theoretic properties. In particular, a mechanism in which every agent has a dominant strategy— an optimal strategy whatever the strategies and preferences of other agents —is useful computationally, in addition to game-theoretically. From a computational perspective, with a dominant strategy an agent can avoid costly modeling and game-theoretic reasoning about other agents.

The *valuation complexity* of a mechanism is related to the amount of *information revelation* that is required from agents, and the complexity of providing that information. Agents must often compute their value for different outcomes, each of which might involve solving a hard *local* optimization problem. This is an important problem that is essentially ignored in classic mechanism design. For example, in a *single-shot direct-revelation* mechanism such as the VCG mechanism, agents must provide *complete information* about their preferences.

Approaches to reduce valuation complexity include:

- Design iterative mechanisms that solve problems with minimal information revelation from agents.
- Provide structured bidding languages to allow compact representations of agent preferences in high-dimensional problems, and exploit the structure computationally throughout the mechanism.

An *iterative mechanism* allows agents to provide incremental information, and can hope to solve the implementation problem *without unnecessary information* about agent preferences.

Of course, to reduce the amount of valuation work required by an agent it is also necessary that an agent can provide this incremental information without first evaluating its complete preferences over all outcomes; i.e. follow its optimal strategy without computing its complete preferences. For example, it is useful if partial orderings over outcomes allow an agent to provide a response.

1.1.4 A Challenge: Incentive-Compatible Iterative Mechanisms

One important challenge in computational mechanism design, given bounded-rational but self-interested agents, and hard combinatorial domains, is:

... develop an iterative mechanism in which incremental revelation of truthful information is a dominant strategy for every agent, and which computes an optimal solution to the system-wide problem with minimal total information revelation.

Addressing this challenge requires a very careful synthesis of ideas from AI, for example to handle high-dimensional bid spaces; from algorithmic theory, for example to solve intermediate allocation and pricing problems; and from game theory to wrap all of this within a strategy-proof system.

The goal is to allow agents to reveal information about their preferences, perhaps approximate and perhaps incomplete, whenever that information is required to compute the optimal system-wide solution and without concern for strategic effects. In this way distributed agent computation on the value for different outcomes can be performed in parallel within a joint search for a system-wide solution, with only as much information computed about the local problems of each agent as is required to compute and verify an solution.

The incentive-engineering problem depends on agent preferences, and the mathematical formulation to make a mechanism incentive-compatible is indepedent of the language with which agents can represent their preferences. However, the computational efficiency of a solution will depend on the representation language; the language should allow an agent to accurately state information about its preferences and/or respond to challenges in an efficient and compact form, and allow tractable computation by the mechanism infrastructure.

This dissertation proposes an iterative combinatorial auction, *i*Bundle Extend&Adjust, that makes significant progress towards this challenge for the *combinatorial allocation problem* (CAP), introduced in the next section. It is important to avoid complete information revelation in solving the combinatorial allocation problem, because agents often have hard valuation problems to compute their value for any single outcome, and there are an exponential number of possible outcomes.

1.2 The Combinatorial Allocation Problem

The combinatorial allocation problem (CAP) is a resource allocation problem in which a set of items are to be allocated across a set of agents. Agents are assumed to have non-linear values for bundles of items, e.g. "I only want A if I also get B", and the goal is to determine the allocation that maximizes the total value over all agents.

The CAP is relevant to many interesting and important real-world applications, including scheduling, logistics and network computation domains. Indeed, it has attracted considerable recent attention in the academic literature because of its application to the FCC spectrum auctions. It is quite reasonable that there are geographical synergies across licenses, for example with the value of a wireless license for the New York metropolitan area contingent on also acquiring a license for the Philadelphia and Boston areas.

Moreover, the classic mechanism design solution to the CAP, the Vickrey-Clarke-Groves mechanism, often requires the solution to a number of intractable problems. Not only must agents report their values for a perhaps exponentially large number of different combinations of items (and solve a perhaps NP-hard problem to compute their value for any one combination), but the mechanism infrastructure must solve a number of NP-hard problems to compute the outcome of the mechanism.

In the CAP there are a set $\mathcal{G}$ of discrete items and a set $\mathcal{I}$ of agents. Each agent has a valuation function $v_i : 2^G \to \mathbb{R}_+$, that defines its value $v_i(S) \geq 0$ for bundles of items, $S \subseteq \mathcal{G}$. The valuation function defines the agent's *type*, i.e. its preferences over different outcomes. Assume $v_i(\emptyset) = 0$, and *free disposal* of items, implying that agents have weakly increasing values for bundles, i.e. $v_i(S) \leq v_i(S')$ for all $S' \supset S$.

An outcome defines an allocation $\mathbf{S} = (S_1, \ldots, S_I)$ of items to agents, with agent i receiving bundle S_i. A feasible allocation assigns each item to no more than one agent. The social choice function, or objective, is allocative-efficiency, i.e. to select the allocation that maximizes the total value over agents.

$$\max_{(S_1,\ldots,S_I)} \sum_{i \in \mathcal{I}} v_i(S_i) \qquad \text{(CAP)}$$
$$\text{s.t.} \quad S_i \cap S_j = \emptyset, \quad \forall i \neq j$$

In words, the CAP problem is to compute an allocation of items to maximize the total

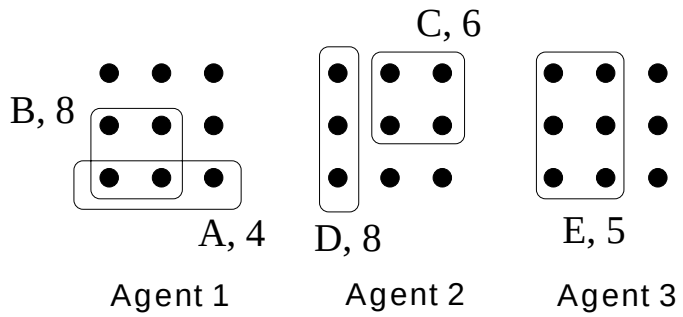


Figure 1.2: An example combinatorial allocation problem.

value over the agents, without assigning the same item to more than one agent.

Of course, in mechanism design we assume that the valuation functions are *private information* to agents, and a solution to the CAP must handle this self-interest, for example providing incentives to promote truth-revelation.

A simple combinatorial allocation problem is illustrated in Figure 1.2. In this example there are nine items (shown as disks) and three agents, with positive values for bundles of items as shown (and zero values for smaller bundles, the same value for any bundle that contains the indicated bundle). For example, agent 1 has value 4 for the bundle identified A, and value 8 for the bundle identified B. The optimal solution, to maximize the total value across all agents, is to allocate bundle A to agent 1 and bundle C to agent 2, with no allocation to agent 3, for total value 10.

1.2.1 Application Domains

The combinatorial allocation problem arises in many domains. Consider the following examples:

- *Dynamic resource allocation* [FNSY96, Che00]. Consider a bandwidth allocation problem. Slots of bandwidth are available, of a fixed size and duration, with network tie-in points. Agents have value for bundles of slots, for example representing a virtual circuit of a particular bandwidth in a particular time interval.

- *Job shop scheduling.* [WWWMM01] Machine time in a flexible manufacturing environment is to be allocated across competing jobs. Jobs require time slots across a sequence of machines, and have deadlines and costs of delay.

- *Course registration.* [GSS93] A number of slots are available in different classes and sections, and each student wants to register for a bundle of classes that fit their major and have no time scheduling conflicts.

- *Airport landing and takeoff scheduling.* [RSB82] A number of take-off and landing slots are available across major airports in the U.S. Competing airlines need pairs of takeoff and landing slots that are compatible with flight-time and schedule requirements.

- *Distributed vehicle routing.* [San93, LOP+00] A number of delivery tasks are to be assigned to a fleet of independent trucks. The marginal cost to pick-up a packet from a location where a drop-off is to be performed is quite small, so the value to an agent for performing one task is contingent on being able to perform other compatible tasks.

- *The FCC Spectrum allocation problem.* [McM94] The FCC was mandated by Congress to achieve an efficient (value-maximizing) allocation of new spectrum license to wireless telephone companies. The mobility of clients leads to synergistic values across geographically consistent license areas, for example the value for New York City, Philadelphia, and Washington DC might be expected to be much greater than the value of any one license by itself.

- *Collaborative planning.* [HG00] Consider a system of robots that want to perform a set of tasks, and have a joint goal to perform the tasks at as low a total cost as possible. Roles in a team to be conditioned on various constraints, for example time constraints, to protect the feasibility of local commitments.

- *Distributed Query Optimization.* [SDK+94] Consider the problem of performing a query that has a number of components. Each agent might have a different expertise, and be able to efficiently answer different decompositions of the query.

- *Supply-chain coordination.* [WWY00] Consider the problem of allocating components to competing manufacturers. Each manufacturer needs a supply of the right combination of components for its own product, without all the components the supply is useless.

- *Travel packages.* [WWO+01] Consider the allocation of flights, hotel rooms, and entertainment tickets to agents that represent clients with different preferences over location, price, hotels, and entertainment. Moreover, a client has no value for an outward flight without a matching return flight or a hotel room without a flight.

1.2.2 The Generalized Vickrey Auction

The Vickrey-Clarke-Groves [Vic61, Cla71, Gro73] provides a dominant-strategy solution to the combinatorial allocation problem. Known as the Generalized Vickrey Auction (or GVA) in this domain, the mechanism is a *one-shot direct-revelation* solution. In the first stage the agents are asked to report their valuation functions. Then, the mechanism computes an allocation based on the information, and computes payments to agents. The payments make truth-revelation the dominant strategy for an agent, whatever the information reported by other agents (and even if other agents are untruthful!). As such, the mechanism successfully aligns the incentives of individual agents with the system-wide objective of computing an efficient allocation.

The VCG mechanism is introduced in detail in the next chapter, but for now let us note that it satisfies the following desirable properties:

Desiderata

- **Allocative efficiency.** The mechanism implements the efficient allocation, the allocation that maximizes total value across agents, given rational agent strategies.

- **Strategy-proofness.** Truth-revelation is optimal for an agent whatever the strategies of other agents, there can be no successful (unilateral) manipulation of the outcome.

- **Individual-rationality.** The expected utility from participation is non-negative (with a rational strategy), whatever the strategies of other agents.

- (Weak) **Budget-balance.** Each agent makes a non-negative payment to the auctioneer, so the net revenue collected by the auctioneer is non-negative.

In fact, in quite a strong sense (see the next chapter) the Vickrey-Clarke-Groves mechanism defines the *only* mechanism for the combinatorial allocation problem with these properties. The mechanism is unique amongst single-shot direct revelation mechanisms, which are sufficient via the revelation principle. However, as discussed above in Section 1.1.3, direct revelation mechanisms are quite unattractive computationally in many interesting application domains. Consider for example the distributed vehicle routing application, in which each truck would need to compute and report its value for all combinations of jobs that it has positive value.

My dissertation develops an *iterative* mechanism, an ascending-price combinatorial auction, with provable efficiency properties for agents that follow a myopic best-response strategy; i.e. taking prices as fixed and bidding for the bundle that maximizes their surplus in each round. I also propose an extended auction that provably terminates with Vickrey payments in a number of interesting special-cases, and an experimental method that is shown empirically to compute Vickrey payments across a complete suite of problems.

An iterative combinatorial auction that terminates with Vickrey payments and the efficient allocation shares many of the good game-theoretic properties of the GVA, i.e. myopic best-response becomes a sequentially-rational equilibrium of the auction, and relaxes the computational demands on agents. An agent can follow a myopic best-response strategy with an approximate evaluation of its preferences over different outcomes, for example with lower- and upper- bounds on its value for different bundles.

1.3 Iterative Combinatorial Auctions

Combinatorial auctions allow agents to bid for bundles of items directly, and express logical constraints over items, such as "I only want A if I also get B." Early combinatorial auctions were proposed to solve distributed optimization problems with self-interested agents that have contingent values for items; e.g. an airport scheduling problem in which planes need *pairs* of compatible takeoff and landing slots, or a course registration problem in which students can bid to take sets of classes. Although combinatorial auctions can be approximated by multiple auctions on single items, this often results in inefficient outcomes and can lead to difficult bidding problems for agents.

In an *iterative* combinatorial auction agents can adjust their bids in response to bids

from other agents, as the auctioneer updates a provisional allocation and bundle prices. Iterative combinatorial auctions can compute optimal solutions to realistic problem instances with less information than sealed-bid auctions, and without agents computing accurate values for all interesting bundles of items. Despite a considerable research effort over the past decade, in both artificial intelligence and economics, it was not known how to design an optimal iterative combinatorial auction in general problems until my dissertation.

*i*Bundle is an ascending-price combinatorial auction, in which agents can adjust their bids for bundles of items across rounds. The auction computes a provisional allocation to maximize revenue in each round, and increases prices on bundles of items based on unsuccessful bids from agents. It terminates as soon as soon as every agent still bidding in the auction wins a set of items in the provisional allocation, with an efficient allocation (maximizing total agent value) for agents that follow a myopic best-response bidding strategy. Myopic best-response assumes that agents bid for bundles that maximize their utility taking the current ask prices as fixed.

The two most important contributions are:

(a) an iterative combinatorial auction that terminates with the efficient allocation for agents that follow myopic best-response bidding strategies.

(b) an iterative combinatorial auction that terminates with the efficient allocation and Vickrey-Clarke-Groves payments, provably in special-cases, and conjectured (with experimental support) in all cases.

With this result the auction inherits much of the strategy-proofness of the VCG solution, and myopic best-response becomes a sequentially-rational strategy for an agent.

The central components of my solution are:

- ***i*Bundle**. I introduce an ascending-price auction, *i*Bundle, which is the first iterative auction for the CAP that implements efficient allocations for a reasonable bidding strategy, in this case with myopic best-response strategies. A myopic best-response strategy is to bid for the bundle that maximizes surplus (value - price), taking the prices in the current round as fixed and ignoring the effect of bids on future prices and future strategies of other agents. *i*Bundle allows incremental information revelation by agents and solves realistic problems without agents revealing, or even computing, complete information about their local preferences.

- **Extend and Adjust.** I propose the *Extend&Adjust* methodology to extend *i*Bundle

for a number of additional rounds, and compute a price discount to each agent at the end of the auction. My conjecture,[1] is that the discounted prices are equal to the payments in the Vickrey-Clarke-Groves (VCG) mechanism for the combinatorial allocation problem. This is significant because it makes myopic best-response a sequentially-rational strategy in equilibrium, inheriting a good degree of robustness-to-manipulation from the VCG mechanism.

- **Proxy Bidding Agents.** Finally, I propose *proxy bidding agents*, that sit between real agents and the auction and restrict agents to myopic best-response strategies for some (perhaps untruthful) valuation function. Agents provide proxy agents with incremental value information, and the proxy agents submit best-response bids whenever there is enough information. The proxies enforce consistency of incremental preference information across rounds. The effect is to further boost robustness-to-manipulation, while retaining useful computational properties.

The combined system, of *i*Bundle Extend&Adjust and proxy bidding agents, provides a quite compelling framework for an iterative and strategy-proof mechanism for the combinatorial allocation problem.

1.3.1 Theoretical Underpinnings

The proof of optimality makes an interesting connection with linear programming theory. *i*Bundle implements a primal-dual algorithm for a linear program formulation of the combinatorial allocation problem. The provisional allocation in each round is a feasible solution to the primal problem, and the ask prices in each round have a natural interpretation as a feasible solution to the dual problem.

Myopic best-response strategies from agents provide enough information to compute and verify primal and dual solutions to the CAP that satisfy complementary slackness conditions. This is the basic methodology that allows optimal solutions to the combinatorial allocation problem to be computed without complete information about agents' valuation functions. Simply announcing a feasible dual solution and computing a primal solution that satisfies complementary slackness conditions is enough. A similar connection

[1] This conjecture is proved in a number of special cases, and a general proof is limited only by the lack of a technical lemma about termination.

was made by Bertsekas for the Assignment problem [Ber87], in which each agent wants at most one item.

Perhaps the most significant technical contribution in this dissertation is to connect this primal-dual auction-based methodology back to the Vickrey-Clarke-Groves mechanism. Although previous authors have recognized that it is useful to compute Vickrey payments at the end of an iterative auction, to the best of my knowledge this is the first direct application of primal-dual techniques to compute Vickrey payments in the general combinatorial allocation problem.

In its basic form, an agent can manipulate the outcome of *i*Bundle, for example placing jump bids, signaling false intentions, or waiting to bid until the end of the auction. A concrete example is provided in Chapter 7. By computing the Vickrey payments at the end of the auction, myopic best-response becomes a sequentially rational strategy for an agent in equilibrium. In addition, with the proxy bidding agents, incremental truth-revelation becomes a *dominant* best-response to *any* value information provided by other agents, so long as that information is not itself conditioned on information revealed dynamically during the auction.

I provide a linear program formulation to compute Vickrey payments, and derive a primal-dual algorithm, VickAuction, to compute the Vickrey payments and efficient allocation. VickAuction computes the outcome of the GVA without complete information revelation from agents, instead requiring that agents provide myopic best-response to a sequence of ascending prices on bundles. VickAuction has a natural interpretation as an iterative combinatorial auction, *i*Bundle Extend&Adjust. The linear program formulation for Vickrey payments computes the minimal "competitive equilibrium" (CE) price on the bundle each agent receives in the efficient allocation. Although there are some problems in which no single set of CE prices support Vickrey payments to every agent, I show that it is always possible to compute the Vickrey payment as the minimal price to each agent over all CE prices.

The approach in *i*Bundle Extend&Adjust is to:

(a) keep the auction open long enough to collect enough best-response information from agents to compute Vickrey payments as the minimal CE prices

(b) adjust prices towards Vickrey payments after the auction terminates, so that agents pay less than what they finally bid.

1.3.2 Approximations and Special-Cases

The winner-determination problem in each round of *i*Bundle remains NP-hard. However, each problem instance in *i*Bundle is typically smaller and easier to solve than the problem instances that an auctioneer must solve in the single-shot direct-revelation Vickrey-Clarke-Groves mechanism. Agents only need to bid for the bundles that maximize their utility in each round of *i*Bundle, while they must bid for all bundles with positive value in the VCG solution.

However, there is no escaping the NP-hardness of the top-level combinatorial allocation problem, and it is necessary to introduce approximations and/or identify special-cases for large problem instances. There are a number of interesting ways to introduce approximations within *i*Bundle without changing the incentives for agents to follow the same myopic best-response strategy.

First, we can increase the minimal bid-increment in the auction, which defines the rate at which prices are increased across rounds. The number of rounds in the auction are approximately inversely-proportional to the bid increment, so doubling the bid increment halves the number of rounds to termination and the number of winner-determination problems to solve. Experimental results demonstrate an order-of-magnitude speed-up over the VCG mechanism with at least 99% allocative efficiency, with the same combinatorial optimization algorithm to solve winner-determination problems in both mechanisms. Additional significant speed-ups are achieved as the bid increment is increased and allocative efficiency is traded for computational efficiency.

Second, we can introduce approximate winner-determination algorithms into *i*Bundle. A simple property of "bid monotonicity" (see Chapter 5) ensures that the same incentives are present for agents to follow a myopic best-response bidding strategy. Experimental analysis demonstrates that the auction can often achieve quite high allocative-efficiency with negligible computation using a greedy winner-determination algorithm.

Third, we can identify tractable special-cases of the winner-determination, and restrict agents to bidding languages that are compatible with these tractable special-cases. In cases in which this restriction leaves agents with enough expressiveness to follow truthful myopic best-response this gives a significant computational speed-up with no loss in allocative-efficiency. In general the expressiveness of the language leads to a tradeoff between computational and allocative efficiency, and can also change the incentive properties

of an auction if an agent is forced to choose from a "second-best" set of bids.

1.3.3 Myopic Best-response vs. Direct Revelation

I would like to comment briefly about the complexity of myopic best-response, in comparison with direct revelation. In addition to solving the CAP with incremental information revelation, it is important that agents can follow myopic best-response strategies without computing their exact value for all bundles.

Consider the simpler case of a single-item allocation problem. Assume that an agent maintains bounds on its value for the item, and has an approximation algorithm that updates the bounds for a cost (perhaps the opportunity cost of lost computation time on another problem). Compare the valuation problem for an agent in an ascending-price auction with that in a sealed-bid auction:

(a) *ascending-price auction.* In the ascending-price auction the agent can bid while the price is less than its lower bound, and drop out when the price is greater than its upper bound. It only needs to perform valuation work when the price is between its two bounds and the auction is about to terminate. Intuitively, if the agent's actual value is close to the highest value over all other agents then it will need to compute a quite accurate value to place optimal bids. However, if the agent's actual value is quite far from the highest outside value then it will be able to bid optimally with a quite rough approximation.

(b) *sealed-bid auction.* In the sealed-bid auction an uninformed agent, that has no information about the bids from the other agents, must compute its exact value to submit an optimal bid. As soon as it bids an approximate value it risks missing a good outcome, if the second highest bid in the auction is between its true value true and its bid price.

The same computational advantages hold for iterative solutions in combinatorial allocation problems. In a combinatorial search space it is important that an agent can prune that space effectively, based on ask prices and its approximate values for bundles, for example using the value of one bundle to make inferences about the values of other bundles, or the price for one bundle to make inferences about the prices of other bundles.

The effect of auction design on allocative-efficiency and agent deliberation is explored in more detail in Chapter 8, with a concrete model of agent valuation and bounded-rationality.

1.4 Bounded Rational Compatible Auctions

To capture the notion of an auction that allows an agent to bid without computing its exact value for all possible outcomes, I define *bounded-rational compatible* auctions. A bounded-rational compatible (BRC) auction allows an agent to compute its equilibrium bidding strategy with only approximate information about its own preferences across outcomes. This parallels the concept of a strategy-proof auction, in which an agent can compute its optimal strategy without modeling the other agents. BRC auctions are useful in application to domains with agents that have limited computation and hard valuation problems.

Iterative auctions present a special case of bounded-rational compatible auctions. As discussed above, in an iterative auction an agent can follow a myopic best-response strategy in response to a sequence of prices with an approximate valuation functions. I say that an iterative auction is *myopic bounded-rational compatible*. By comparison, a strategy-proof single-shot sealed-bid auction, in which truth-revelation is an agent's dominant strategy, is not (dominant-strategy) bounded-rational compatible because an agent cannot reveal complete information about its valuation function with an approximate valuation function.

A bounded-rational compatible auction need not be allocatively-efficient. For example, a posted-price auction is bounded-rational compatible, but not efficient unless the auctioneer is well-informed about the preferences of agents and able to set equilibrium prices. Experimental results for a simple model of bounded-rational agents, with costly and/or limited computation, demonstrate that iterative auctions can compute more efficient allocations than sealed-bid auctions. Feedback in an iterative auction allows agents to focus limited computational resources on computing values for those outcomes that are important to find a good fit with the preferences of other agents. In addition to leading to more efficient allocations, the results also demonstrate that it is possible to reduce the total amount of computation that agents must perform to compute an efficient allocation.

1.5 Important Related Work

I introduce related literature throughout my dissertation, in appropriate places. However, let me briefly mention some work that is very closely related in motivation and methodology.

Nisan & Ronen [NR00, NR01] have been able to make some connections between approximate algorithms, bounded-rational agents, and mechanism design. This work, that they coin *algorithmic mechanism design*, considers the following aspects:
(a) connections between approximate winner-determination algorithms and approximate mechanisms
(b) a "challenge and response" mechanism to improve incentive-compatibility with approximate winner-determination algorithms
(c) a "feasible dominance" concept, that describes an auction in which truth revelation is optimal given a subset of possible strategies.

One thing that is absent from their work is any consideration of the costs of agent valuation and preference revelation. The focus instead is on the cost of winner-determination and strategic behavior. Similarly, the work of Tennenholtz *et al.* [TKDM00] on the properties required for an approximate winner-determination algorithm to satisfy incentive-compatibility within a Vickrey like mechanism focuses in on single-shot sealed-bid mechanisms.

Nisan [Nis00] has also considered the tradeoffs between the expressiveness and efficiency of bidding languages, and proposed a language OR* with a good combination of properties. Recently La Mura and Shoham [MS99] have considered the role of concise bidding languages in auctions, for example using hierarchical tree structures to represent preferences and reasoning in this abstract representation language.

Shoham & Tennenholtz's [ST01] work on the communication complexity of auction mechanisms does draw comparisons between iterative and single-shot sealed-bid auctions. For example, the authors note that a Dutch (descending-price auction) is a minimal computational complexity solution to a single-item allocation problem. This work does not consider the valuation work required of agents, just the communication complexity.

Feigenbaum *et al.* [FPS00] have proposed a decentralized mechanism design for routing in a multi-cast tree, with self-interested users at the nodes interested in receiving optimal streams of information. The mechanism design is innovative in that it decentralizes the winner-determination work to nodes in the tree.

Wurman & Wellman propose a design for an iterative combinatorial auction, AkBA [Wur99, WW00]. The main differences between AkBA and iBundle are in the structure of prices and price updates. AkBA has only anonymous prices, and although it terminates in

an equilibrium (for agents, but not the auctioneer), the cost is a loss in allocative efficiency in some CAP problem instances. Nevertheless, experimental results demonstrate high allocative efficiency for myopic best-response agents across a set of problems. An analysis of the incentive properties of minimal and maximal dual price solutions is also performed, but no strong connection is made to Vickrey-Clarke-Groves payments.

Other important work in iterative auction design includes that of Gul & Stacchetti [GS00], and particularly Ausubel [Aus97, Aus00]. Ausubel is able to achieve Vickrey payments even in some problems in which Vickrey payments are supported in no competitive equilibrium. Recent analysis by Bikchandani *et al.* [BdVSV01] provides a primal-dual interpretation of Ausubel's methods. A primal-dual approach was earlier proposed for the Assignment problem, by Leonard [Leo83] and Demange et al. [DGS86]. Bikchandani *et al.* [BdVSV01] and Bikchandani & Ostroy [BO00] also discuss a linear programming model for computing VCG payments in the general CAP problem. However I am not aware of any earlier work that develops a primal-dual method, or an auction mechanism, to compute the Vickrey payments in the general problem.

Sandholm [San93, SL95, SL96, SL97, San00] has proposed a number of methods to handle agent bounded-rationality and game-theoretic concerns in a distributed system. In contrast to my work on auction mechanisms, much of Sandholm's work relates to decentralized task/resource allocation systems with no centralized auctioneer. Some contributions that are relevant to my work include:

(1) a marginal-cost analysis of a contract-net style protocol, that demonstrates how agents might bid with approximate solutions to local valuation problems.

(2) a *leveled-commitment* protocol that allows agents to make initial contracts under uncertainty and decommit if necessary at a later time.

(3) a mechanism to allow coalition formation between bounded-rational self-interested agents, that allows explicitly for the cost of computing the values of different coalitions during negotiation.

In recent work, Larson & Sandholm [LS00] have studied a model of deliberation in equilibrium, for bounded-rational agents in a strategic environment.

1.6 Outline

Chapter 2 introduces important ideas from game theory, economics, and mechanism design, including concepts of Nash equilibrium and dominant strategy equilibrium, the *revelation principle*, the Vickrey-Clarke-Groves family of mechanisms, and impossibility and possibility results.

Chapter 3 introduces concerns in computational mechanism design, and considers a number of ways to handle computational complexity, at both the agent and the infrastructure level.

Chapters 4 and 5 relate to the design of an iterative auction under the assumption that agents follow myopic best-response bidding strategies, while Chapters 6 and 7 relate to an extension to the design to justify myopic best-response with a connection back to the Vickrey-Clarke-Groves mechanism.

Chapter 4 introduces a linear programming approach to iterative mechanism design, and presents a primal-dual algorithm CombAuction for the combinatorial allocations problem. The algorithm corresponds with *i*Bundle for myopic best-response agent strategies. Chapter 5 also surveys the known tractable special-cases of the combinatorial allocation problem. Chapter 4 also contains a survey of earlier iterative auction designs both (implicitly at least) primal-dual based, and more ad-hoc. Chapter 5 presents *i*Bundle, my ascending-price combinatorial auction, along with experimental results relating to both the economic and computational efficiency of the auction.

Chapter 6 introduces a linear programming approach to computing Vickrey payments in the combinatorial allocation problem. A novel linear programming formulation is derived to compute Vickrey payments from a suitable set of competitive equilibrium prices and the efficient allocation. This leads to an adjustment procedure Adjust* to take prices at the end of *i*Bundle and compute discounts to each agent, adjusting prices towards Vickrey payments. I characterize necessary and sufficient conditions for the procedure to compute Vickrey payments, and propose a complete primal-dual algorithm, VickAuction to compute Vickrey payments. This is significant, because it has a natural auction interpretation as an extension to *i*Bundle.

Chapter 7 presents the Extend&Adjust method, which introduces a second phase to *i*Bundle. The purpose of the second phase is to collect enough additional information from agents to compute Vickrey payments when the auction terminations, in addition

to the efficient allocation. Proxy bidding agents are introduced, to boost robustness-to-manipulation in an iterative auction that terminates with Vickrey payments with myopic best-response agent bidding strategies. Experimental results demonstrate the convergence of auction prices in *i*Bundle Extend&Adjust to Vickrey payments in general CAP problems. Vickrey payments make myopic best-response a Bayesian-Nash equilibrium of the auction.

Chapter 8 focuses on agent computation in mechanisms, in particular on the agent valuation work that is required across different solutions. *Bounded-rational compatibility* is proposed to characterize auctions that allow an agent to compute an optimal strategy with approximate information about its own preferences. Experimental results compare the efficiency and agent computation in iterative and sealed-bid auctions, for a simple model of agent bounded-rationality and myopic metareasoning. I also present a *structural analysis* of the worst-case complexity of myopic best-response in combinatorial auctions, in comparison with the worst-case complexity of complete revelation, and identify conditions that provide an exponential speed-up for iterative mechanisms.

Chapter 9 considers a concrete application of an auction method to a distributed optimization problem, a distributed train scheduling problem. The problem is to compute a shared schedule for trains over a network, such that trains run as close to on-time as possible and the cost of early/late arrivals and departures is minimized. The structural assumption is that trains are self-interested and autonomous, and would like to travel on-time irrespective of the effect on other trains. In addition, individual dispatcher agents control separate network territories. Trains bid for the right to enter and exit territories at particular times, and dispatchers select bids to maximize revenue such that a safe schedule exists to get trains across the region. The continuous quality of time is handled in the auction with a constraint-based bidding language. The train-scheduling problem is not a combinatorial allocation problem, for example because the feasible combinations of items (which we can think of as entry and exit times) are not statically defined, but depend on the ability to construct feasible time-location schedules for trains. However, *i*Bundle price-update, winner-determination and termination semantics prove useful in this domain.

My conclusions are in Chapter 10, together with a brief discussion of interesting areas for future work.

Chapter 2

Classic Mechanism Design

Mechanism design is the sub-field of microeconomics and game theory that considers how to implement good system-wide solutions to problems that involve multiple self-interested agents, each with private information about their preferences. In recent years mechanism design has found many important applications; e.g., in electronic market design, in distributed scheduling problems, and in combinatorial resource allocation problems.

This chapter provides an introduction to the the game-theoretic approach to mechanism design, and presents important possibility and impossibility results in the literature. There is a well-understood sense of what can and cannot be achieved, at least with fully rational agents and without computational limitations. The next chapter discusses the emerging field of *computational* mechanism design, and also surveys the economic literature on limited communication and agent bounded-rationality in mechanism design. The challenge in computational mechanism design is to design mechanisms that are *both* tractable (for agents and the auctioneer) and retain useful game-theoretic properties. For a more general introduction to the mechanism design literature, MasColell *et al.* [MCWG95] provides a good reference. Varian [Var95] provides a gentle introduction to the role of mechanism design in systems of computational agents.

In a mechanism design problem one can imagine that each agent holds one of the "inputs" to a well-formulated but incompletely specified optimization problem, perhaps a constraint or an objective function coefficient, and that the system-wide goal is to solve the specific instantiation of the optimization problem specified by the inputs. Consider for example a network routing problem in which the system-wide goal is to allocate resources to minimize the total cost of delay over all agents, but each agent has private information about parameters such as message size and its unit-cost of delay. A typical approach

in mechanism design is to provide incentives (for example with suitable payments) to promote truth-revelation from agents, such that an optimal solution can be computed to the distributed optimization problem.

Groves mechanisms [Gro73] have a central role in classic mechanism design, and promise to remain very important in computational mechanism design. Indeed, Groves mechanisms have a focal role in my dissertation, providing strong guidance for the design of mechanisms in the combinatorial allocation problem. Groves mechanisms solve problems in which the goal is to select an outcome, from a set of discrete outcomes, that maximizes the *total value* over all agents. The Groves mechanisms are *strategy-proof*, which means that truth-revelation of preferences over outcomes is a dominant strategy for each agent— optimal whatever the strategies and preferences of other agents. In addition to providing a robust solution concept, strategy-proofness removes game-theoretic complexity from each individual agent's decision problem; an agent can compute its optimal strategy without needing to model the other agents in the system. In fact (see Section 2.4), Groves mechanisms are the *only* strategy-proof and value-maximizing (or efficient) mechanisms amongst an important class of mechanisms.

But Groves mechanisms have quite bad computational properties. Agents must report complete information about their preferences to the mechanism, and the optimization problem— to maximize value —is solved centrally once all this information is reported. Groves mechanisms provide a completely centralized solution to a decentralized problem. In addition to difficult issues such as privacy of information, trust, etc. the approach fails computationally in combinatorial domains either when agents cannot compute their values for all possible outcomes, or when the mechanism cannot solve the centralized problem. Computational approaches attempt to retain the useful game-theoretic properties but relax the requirement of complete information revelation. As one introduces alternative distributed implementations it is important to consider effects on game-theoretic properties, for example the effect on strategy-proofness.

Here is an outline of the chapter. Section 2.1 presents a brief introduction to game theory, introducing the most important solution concepts. Section 2.2 introduces the theory of mechanism design, and defines desirable mechanism properties such as efficiency, strategy-proofness, individual-rationality, and budget-balance. Section 2.3 describes the revelation principle, which has proved a powerful concept in mechanism design theory, and

introduces incentive-compatibility and direct-revelation mechanisms. Section 2.4 presents the efficient and strategy-proof Vickrey-Clarke-Groves mechanisms, including the Generalized Vickrey Auction for the combinatorial allocation problem. Sections 2.5 and 2.6 summarize the central impossibility and possibility results in mechanism design. Finally, Section 2.7 is provides a brief discussion of optimal auction design and the conflict between the goals of revenue-maximization and efficiency.

2.1 A Brief Introduction to Game Theory

Game theory [vNM47, Nas50] is a method to study a system of self-interested agents in conditions of strategic interaction. This section provides a brief tour of important game-theoretic solution concepts. Fudenberg & Tirole [FT91] and Osborne & Rubinstein [OR94] provide useful introductions to the subject. Places to start for auction theory include McAfee & McMillan [PMM87] and Wurman *et al.* [WWW00].

2.1.1 Basic Definitions

It is useful to introduce the idea of the *type* of an agent, which determines the preferences of an agent over different outcomes of a game. This will bring clarity when we discuss mechanism design in the next section. Let $\theta_i \in \Theta_i$ denote the type of agent i, from a set of possible types Θ_i. An agent's preferences over outcomes $o \in \mathcal{O}$, for a set $\mathcal{O}$ of outcomes, can then be expressed in terms of a utility function that is parameterized on the type. Let $u_i(o, \theta_i)$ denote the utility of agent i for outcome $o \in \mathcal{O}$ given type θ_i. Agent i prefers outcome o_1 over o_2 when $u_i(o_1, \theta_i) > u_i(o_2, \theta_i)$.

The fundamental concept of agent choice in game theory is expressed as a *strategy*. Without providing unnecessary structure, a strategy can loosely be defined as:

DEFINITION 2.1 [strategy] A strategy is a complete contingent plan, or decision rule, that defines the action an agent will select in every distinguishable state of the world.

Let $s_i(\theta_i) \in \Sigma_i$ denote the strategy of agent i given type θ_i, where Σ_i is the set of all possible strategies available to an agent. Sometimes the conditioning on an agent's type is left implicit, and I write s_i for the strategy selected by agent i given its type.

In addition to *pure*, or deterministic strategies, agent strategies can also be *mixed*, or stochastic. A mixed strategy, written $\sigma_i \in \Delta(\Sigma_i)$ defines a probability distribution over

pure strategies.

Example. In a single-item ascending-price auction, the state of the world (p, x) defines the current ask price $p \geq 0$ and whether or not the agent is holding the item in the provisional allocation $x \in \{0, 1\}$. A strategy defines the bid $b(p, x, v)$ that an agent will place for every state, (p, x) and for every value $v \geq 0$ it might have for the item. A best-response strategy is as follows:

$$b_{\text{BR}}(p, x, v) = \begin{cases} p & \text{, if } x = 0 \text{ and } p < v \\ \text{no bid} & \text{, otherwise} \end{cases}$$

One can imagine that a *game* defines the set of actions available to an agent (e.g. valid bids, legal moves, etc.) and a mapping from agent strategies to an outcome (e.g. the agent with highest bid at the end of the auction wins the item and pays that price, checkmate to win the game, etc.)

Again, avoiding unnecessary detail, given a game (e.g. an auction, chess, etc.) we can express an agent's utility as a function of the strategies of all the agents to capture the essential concept of strategic interdependence.

DEFINITION 2.2 [utility in a game] Let $u_i(s_1, \ldots, s_I, \theta_i)$ denote the utility of agent i at the outcome of the game, given preferences θ_i and strategies profile $s = (s_1, \ldots, s_I)$ selected by each agent.

In other words, the utility, $u_i(\cdot)$, of agent i determines its preferences over its own strategy and the strategies of other agents, given its type θ_i, which determines its base preferences over different outcomes in the world, e.g. over different allocations and payments.

Example. In a single-item ascending-price auction, if agent 2 has value $v_2 = 10$ for the item and follows strategy $b_{\text{BR},2}(p, x, v_2)$ defined above, and agent 1 has value v_1 and follows strategy $b_{\text{BR},1}(p, x, v_1)$, then the utility to agent 1 is:

$$u_1\left(b_{\text{BR},1}(p, x, v_1), b_{\text{BR},2}(p, x, 10), 10\right) = \begin{cases} v_1 \Leftrightarrow (10 + \epsilon) & \text{, if } v_1 > 10 \\ 0 & \text{, otherwise} \end{cases}$$

where $\epsilon > 0$ is the minimal bid increment in the auction and agent i's utility given value v_i and price p is $u_i = v_i \Leftrightarrow p$, i.e. equal to its surplus.

The basic model of agent rationality in game theory is that of an *expected utility maximizer*. An agent will select a strategy that maximizes its expected utility, given its preferences θ_i over outcomes, beliefs about the strategies of other agents, and structure of the game.

2.1.2 Solution Concepts

Game theory provides a number of solution concepts to compute the outcome of a game with self-interested agents, given assumptions about agent preferences, rationality, and information available to agents about each other.

The most well-known concept is that of a Nash equilibrium [Nas50], which states that in equilibrium every agent will select a utility-maximizing strategy given the strategy of every other agent. It is useful to introduce notation $s = (s_1, \ldots, s_I)$ for the joint strategies of all agents, or *strategy profile*, and $s_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1}, s_I)$ for the strategy of every agent except agent i. Similarly, let θ_{-i} denote the type of every agent except i.

DEFINITION 2.3 [Nash equilibrium] A strategy profile $s = (s_1, \ldots, s_I)$ is in Nash equilibrium if every agent maximizes its expected utility, for every i,

$$u_i(s_i(\theta_i), s_{-i}(\theta_{-i}), \theta_i) \geq u_i(s_i'(\theta_i), s_{-i}(\theta_{-i}), \theta_i), \qquad \text{for all } s_i' \neq s_i$$

In words, every agent maximizes its utility with strategy s_i, given its preferences and the strategy of every other agents. This definition can be extended in a straightforward way to include mixed strategies.

Although the Nash solution concept is fundamental to game theory, it makes very strong assumptions about agents' information and beliefs about other agents, and also loses power in games with *multiple* equilibria. To play a Nash equilibrium in a one-shot game every agent must have perfect information (and know every other agent has the same information, etc., i.e. common knowledge) about the preferences of every other agent, agent rationality must also be common knowledge, and agents must all select the same Nash equilibrium.

A stronger solution concept is a *dominant strategy* equilibrium. In a dominant strategy equilibrium every agent has the same utility-maximizing strategy, for all strategies of other agents.

DEFINITION 2.4 [Dominant-strategy] Strategy s_i is a dominant strategy if it (weakly) maximizes the agent's expected utility for all possible strategies of other agents,

$$u_i(s_i, s_{-i}, \theta_i) \geq u_i(s'_i, s_{-i}, \theta_i), \qquad \text{for all } s'_i \neq s_i,\ s_{-i} \in \Sigma_{-i}$$

In other words, a strategy s_i is a dominant strategy for an agent with preferences θ_i if it maximizes expected utility, whatever the strategies of other agents.

Example. In a sealed-bid second-price (Vickrey auction), the item is sold to the highest bidder for the second-highest price. Given value v_i, bidding strategy

$$b_i(v_i) = v_i$$

is a dominant strategy for agent i because its utility is

$$u_i(b_i, b', v_i) = \begin{cases} v_i \Leftrightarrow b' & \text{, if } b_i > b' \\ 0 & \text{otherwise} \end{cases}$$

for bid b_i and highest bid from another agent b'. By case analysis, when $b' < v_i$ then any bid $b_i \geq b'$ is optimal, and when $b' \geq v_i$ then any bid $b_i < b'$ is optimal. Bid $b_i = v_i$ solves both cases.

Dominant-strategy equilibrium is a very robust solution concept, because it makes no assumptions about the information available to agents about each other, and does not require an agent to believe that other agents will behave rationally to select its own optimal strategy. In the context of mechanism design, dominant strategy implementations of social choice functions are much more desirable than Nash implementations (which in the context of the informational assumptions at the core of mechanism design are essentially useless).

A third solution concept is *Bayesian-Nash equilibrium.* In a Bayesian-Nash equilibrium every agent is assumed to share a common *prior* about the distribution of agent types, $F(\theta)$, such that for any particular game the agent profiles are distributed according to $F(\theta)$. In equilibrium every agent selects a strategy to maximize expected utility in equilibrium with expected-utility maximizing strategies of other agents.

DEFINITION 2.5 [Bayesian-Nash] A strategy profile $s = (s_1(\cdot), \ldots, s_I(\cdot))$ is in Bayesian-Nash equilibrium if for every agent i and all preferences $\theta_i \in \Theta_i$

$$u_i(s_i(\theta_i), s_{-i}(\cdot), \theta_i) \geq u_i(s'_i(\theta_i), s_{-i}(\cdot), \theta_i), \qquad \text{for all } s'_i(\cdot) \neq s_i(\cdot)$$

where u_i is used here to denote the *expected* utility over distribution $F(\theta)$ of types.

Comparing Bayesian-Nash with Nash equilibrium, the key difference is that agent i's strategy $s_i(\theta_i)$ must be a best-response to the *distribution* over strategies of other agents, given distributional information about the preferences of other agents. Agent i does not necessarily play a best-response to the *actual* strategies of the other agents.

Bayesian-Nash makes more reasonable assumptions about agent information than Nash, but is a weaker solution concept than dominant strategy equilibrium. Remaining problems with Bayesian-Nash include the existence of multiple equilibria, information asymmetries, and rationality assumptions, including common-knowledge of rationality.

The solution concepts, of Nash, dominant-strategy, and Bayesian-Nash, hold in both *static* and *dynamic* games. In a static game every agent commits to its strategy simultaneously (think of a sealed-bid auction for a simple example). In a dynamic game actions are interleaved with observation and agents can learn information about the preferences of other agents during the course of the game (think of an iterative auction, or stages in a negotiation). Additional refinements to these solution concepts have been proposed to solve dynamic games, for example to enforce *sequential rationality* (backwards induction) and to remove *non-credible threats* off the equilibrium path. One such refinement is subgame perfect Nash, another is perfect Bayesian-Nash (which applies to dynamic games of incomplete information), see [FT91] for more details.

Looking ahead to mechanism design, an ideal mechanism provides agents with a dominant strategy *and* also implements a solution to the multi-agent distributed optimization problem. We can state the following preference ordering across implementation concepts: dominant $\succ$ Bayesian-Nash $\succ$ Nash. In fact, a Nash solution concept in the context of a mechanism design problem is essentially useless unless agents are very well-informed about each others' preferences, in which case it is surprising that the mechanism infrastructure itself is not also well-informed.

2.2 Mechanism Design: Important Concepts

The mechanism design problem is to implement an optimal system-wide solution to a decentralized optimization problem with self-interested agents with private information about their preferences for different outcomes.

Recall the concept of an agent's *type*, $\theta_i \in \Theta_i$, which determines its preferences over different outcomes; i.e. $u_i(o, \theta_i)$ is the utility of agent i with type θ_i for outcome $o \in \mathcal{O}$.

The system-wide goal in mechanism design is defined with a *social choice function*, which selects the optimal outcome given agent types.

DEFINITION 2.6 [Social choice function] Social choice function $f : \Theta_1 \times \ldots \times \Theta_I \rightarrow \mathcal{O}$ chooses an outcome $f(\theta) \in \mathcal{O}$, given types $\theta = (\theta_1, \ldots, \theta_I)$.

In other words, given agent types $\theta = (\theta_1, \ldots, \theta_I)$, we would like to choose outcome $f(\theta)$. The mechanism design problem is to implement "rules of a game", for example defining possible strategies and the method used to select an outcome based on agent strategies, to implement the solution to the social choice function despite agent's self-interest.

DEFINITION 2.7 [mechanism] A mechanism $\mathcal{M} = (\Sigma_1, \ldots, \Sigma_I, g(\cdot))$ defines the set of strategies Σ_i available to each agent, and an *outcome rule* $g : \Sigma_1 \times \ldots \times \Sigma_I \rightarrow \mathcal{O}$, such that $g(s)$ is the outcome implemented by the mechanism for strategy profile $s = (s_1, \ldots, s_I)$.

In words, a mechanism defines the strategies available (e.g., bid at least the ask price, etc.) and the method used to select the final outcome based on agent strategies (e.g., the price increases until only one agent bids, then the item is sold to that agent for its bid price).

Game theory is used to analyze the outcome of a mechanism. Given mechanism $\mathcal{M}$ with outcome function $g(\cdot)$, we say that a mechanism *implements* social choice function $f(\theta)$ if the outcome computed with *equilibrium* agent strategies is a solution to the social choice function for all possible agent preferences.

DEFINITION 2.8 [mechanism implementation] Mechanism $\mathcal{M} = (\Sigma_1, \ldots, \Sigma_I, g(\cdot))$ *implements* social choice function $f(\theta)$ if $g(s_1^*(\theta_1), \ldots, s_I^*(\theta_I)) = f(\theta)$, for all $(\theta_1, \ldots, \theta_I) \in \Theta_1 \times \ldots \times \Theta_I$, where strategy profile $(s_1^*, \ldots, s_I^*)$ is an equilibrium solution to the game induced by $\mathcal{M}$.

The equilibrium concept is deliberately left undefined at this stage, but may be Nash, Bayesian-Nash, dominant- or some other concept; generally as strong a solution concept as possible.

To understand why the mechanism design problem is difficult, consider a very naive mechanism, and suppose that the system-wide goal is to implement social choice function

$f(\theta)$. The mechanism asks agents to *report their types*, and then simply implements the solution to the social choice function that corresponds with their reports, i.e. the outcome rule is equivalent to the social choice function, $g(\hat{\theta}) = f(\hat{\theta})$ given reported types $\hat{\theta} = (\hat{\theta}_1, \ldots, \hat{\theta}_I)$. But, there is no reason for agents to report their true types! In a Bayesian-Nash equilibrium each agent will choose to announce a type $\hat{\theta}_i$ to maximize its expected utility, and solve:

$$\max_{\theta_i' \in \Theta_i} \mathrm{E}_{\theta_{-i}} u_i(\theta_i', s_{-i}(\theta_{-i}), \theta_i)$$

given distributional information about the types of other agents, and under the assumption that the other agents are also following expected-utility maximizing strategies. This announced type $\hat{\theta}_i$ need not equal the agent's true type.

Looking ahead, the mechanism design problem is to design a mechanism— a set of possible agent strategies and an outcome rule —to implement a social choice function with desirable properties, in as strong a solution concept as possible; i.e. dominant is preferred to Bayesian-Nash because it makes less assumptions about agents.

2.2.1 Properties of Social Choice Functions

Many properties of a mechanism are stated in terms of the properties of the social choice function that the mechanism implements. A good place to start is to outline a number of desirable properties for social choice functions.

A social choice function is *Pareto optimal* (or Pareto efficient) if it implements outcomes for which no alternative outcome is strongly preferred by at least one agent, and weakly preferred by all other agents.

Definition 2.9 [Pareto optimal] Social choice function $f(\theta)$ is Pareto optimal if for every $o' \neq f(\theta)$, and all types $\theta = (\theta_1, \ldots, \theta_I)$,

$$u_i(o', \theta_i) > u_i(o, \theta_i) \quad \Rightarrow \quad \exists j \in \mathcal{I} \;\; u_j(o', \theta_j) < u_j(o, \theta_j)$$

In other words, in a Pareto optimal solution no agent can every be made happier without making at least one other agent less happy.

A very common assumption in auction theory and mechanism design, and one which I will follow in my dissertation, is that agents are *risk neutral* and have *quasi-linear utility* functions.

DEFINITION 2.10 [Quasi-linear Preferences] A quasi-linear utility function for agent i with type θ_i is of the form:

$$u_i(o, \theta_i) = v_i(x, \theta_i) \Leftrightarrow p_i$$

where outcome o defines a choice $x \in \mathcal{K}$ from a discrete choice set and a *payment* p_i by the agent.

The type of an agent with quasi-linear preferences defines its *valuation function*, $v_i(x)$, i.e. its value for each choice $x \in \mathcal{K}$. In an allocation problem the alternatives $\mathcal{K}$ represent allocations, and the transfers represent payments to the auctioneer. Quasi-linear preferences make it straightforward to transfer utility across agents, via side-payments.

Example. In an auction for a single-item, the outcome defines the allocation, i.e. which agent gets the item, and the payments of each agent. Assuming that agent i has *value* $v_i = \$10$ for the item, then its utility for an outcome in which it is allocated the item is $u_i = v_i \Leftrightarrow p = 10 \Leftrightarrow p$, and the agent has positive utility for the outcome so long as $p < \$10$.

Risk neutrality follows because an expected utility maximizing agent will pay as much as the expected value of an item. For example in a situation in which it will receive the item with value \$10 with probability π, an agent would be happy to pay as much as $\$10\pi$ for the item.

With quasi-linear agent preferences we can separate the outcome of a social choice function into a choice $x(\theta) \in \mathcal{K}$ and a payment $p_i(\theta)$ made by each agent i:

$$f(\theta) = (x(\theta), p_1(\theta), \ldots, p_I(\theta))$$

for preferences $\theta = (\theta_1, \ldots, \theta_I)$.

The *outcome rule*, $g(s)$, in a mechanism with quasi-linear agent preferences, is decomposed into a *choice rule*, $k(s)$, that selects a choice from the choice set given strategy profile s, and a payment rule $t_i(s)$ that selects a payment for agent i based on strategy profile s.

DEFINITION 2.11 [quasi-linear mechanism] A quasi-linear mechanism $\mathcal{M} = (\Sigma_1, \ldots, \Sigma_I, k(\cdot), t_1(\cdot), \ldots, t_I(\cdot))$ defines: the set of strategies Σ_i available to each agent; a *choice rule* $k : \Sigma_1 \times \ldots \times \Sigma_I \to \mathcal{K}$, such that $k(s)$ is the choice implemented for strategy profile $s = (s_1, \ldots, s_I)$; and *transfer rules* $t_i : \Sigma_1 \times \ldots \times \Sigma_I \to \mathbb{R}$, one for each agent i, to compute the payment $t_i(s)$ made by agent i.

Properties of social choice functions implemented by a mechanism can now be stated *separately*, for both the choice selected and the payments.

A social choice function is *efficient* if:

DEFINITION 2.12 [allocative efficiency] Social choice function $f(\theta) = (x(\theta), p(\theta))$ is *allocatively-efficient* if for all preferences $\theta = (\theta_1, \ldots, \theta_I)$

$$\sum_{i=1}^{I} v_i(x(\theta), \theta_i) \geq \sum_{i} v_i(x', \theta_i), \qquad \text{for all } x' \in \mathcal{K} \tag{Eff}$$

It is common to state this as *allocative* efficiency, because the choice sets often define an allocation of items to agents. An efficient allocation maximizes the total value over all agents.

A social choice function is *budget-balanced* if:

DEFINITION 2.13 [budget-balance] Social choice function $f(\theta) = (x(\theta), p(\theta))$ is *budget-balanced* if for all preferences $\theta = (\theta_1, \ldots, \theta_I)$

$$\sum_{i=1}^{I} p_i(\theta) = 0 \tag{BB}$$

In other words, there are no net transfers out of the system or into the system. Taken together, allocative efficiency and budget-balance imply Pareto optimality.

A social-choice function is *weak* budget-balanced if:

DEFINITION 2.14 [weak budget-balance] Social choice function $f(\theta) = (x(\theta), p(\theta))$ is *weakly budget-balanced* if for all preferences $\theta = (\theta_1, \ldots, \theta_I)$

$$\sum_{i=1}^{I} p_i(\theta) \geq 0 \tag{WBB}$$

In other words, there can be a net payment made from agents to the mechanism, but no net payment from the mechanism to the agents.

2.2.2 Properties of Mechanisms

Finally, we can define desirable properties of mechanisms. In describing the properties of a mechanism one must state: the *solution concept*, e.g. Bayesian-Nash, dominant, etc.; and the *domain of agent preferences*, e.g. quasi-linear, monotonic, etc.

The definitions follow quite naturally from the concept of implementation (see definition 2.8) and properties of social choice functions. A mechanism has property P if it implements a social choice function with property P.

For example, consider the definition of a *Pareto optimal* mechanism:

DEFINITION 2.15 [Pareto optimal mechanism] Mechanism $\mathcal{M}$ is Pareto optimal if it *implements* a Pareto optimal social choice function $f(\theta)$.

Technically, this is *ex post* Pareto optimality; i.e. the outcome is Pareto optimal for the specific agent types. A weaker form of Pareto optimality is *ex ante*, in which there is no outcome that at least one agent strictly prefers and all other agents weakly prefer *in expectation*.

Similarly, a mechanism is efficient if it selects the choice $x(\theta) \in \mathcal{K}$ that maximizes total value:

DEFINITION 2.16 [efficient mechanism] Mechanism $\mathcal{M}$ is efficient if it *implements* an allocatively-efficient social choice function $f(\theta)$.

Corresponding definitions follow for budget-balance and weak budget-balance. In the case of budget-balance it is important to make a careful distinction between *ex ante* and *ex post* budget balance.

DEFINITION 2.17 [*ex ante* BB] Mechanism $\mathcal{M}$ is *ex ante* budget-balanced if the equilibrium net transfers to the mechanism are balanced *in expectation* for a distribution over agent preferences.

DEFINITION 2.18 [*ex post* BB] Mechanism $\mathcal{M}$ is *ex post* budget-balanced if the equilibrium net transfers to the mechanism are non-negative *for all* agent preferences, i.e. every time.

Another important property of a mechanism is *individual-rationality*, sometimes known as "voluntary participation" constraints, which allows for the idea that an agent is often not forced to participate in a mechanism but can decide whether or not to participate. Essentially, individual-rationality places constraints on the *level* of expected utility that an agent receives from participation.

Let $\overline{u}_i(\theta_i)$ denote the expected utility achieved by agent i outside of the mechanism, when its type is θ_i. The most natural definition of individual-rationality (IR) is *interim* IR, which states that the expected utility to an agent that *knows its own preferences but*

has only distributional information about the preferences of the other agents is at least its expected outside utility.

DEFINITION 2.19 [individual rationality] A mechanism $\mathcal{M}$ is (*interim*) individual-rational if for all preferences θ_i it implements a social choice function $f(\theta)$ with

$$u_i(f(\theta_i, \theta_{-i})) \geq \overline{u}_i(\theta_i) \qquad \text{(IR)}$$

where $u_i(f(\theta_i, \theta_{-i}))$ is the *expected utility* for agent i at the outcome, given distributional information about the preferences θ_{-i} of other agents, and $\overline{u}_i(\theta_i)$ is the expected utility for non-participation.

In other words, a mechanism is individual-rational if an agent can always achieve as much expected utility from participation as without participation, given prior beliefs about the preferences of other agents.

In a mechanism in which an agent can withdraw once it learns the outcome *ex post* IR is more appropriate, in which the agent's expected utility from participation must be at least its best outside utility *for all* possible types of agents in the system. In a mechanism in which an agent must choose to participate before it even knows its own preferences then *ex ante* IR is appropriate; *ex ante* IR states that the agent's expected utility in the mechanism, averaged over all possible preferences, must be at least its expected utility without participating, also averaged over all possible preferences.

One last important mechanism property, defined for *direct-revelation* mechanisms, is *incentive-compatibility*. The concept of incentive compatibility and direct-revelation mechanisms is very important in mechanism design, and discussed in the next section in the context of the *revelation principle*.

2.3 The Revelation Principle, Incentive-Compatibility, and Direct-Revelation

The *revelation principle* states that under quite weak conditions any mechanism can be transformed into an equivalent *incentive-compatible direct-revelation mechanism*, such that it implements the same social-choice function. This proves to be a powerful theoretic tool, leading to the central possibility and impossibility results of mechanism design.

A direct-revelation mechanism is a mechanism in which the only actions available to

agents are to make direct claims about their preferences to the mechanism. An incentive-compatible mechanism is a direct-revelation mechanism in which agents report *truthful information* about their preferences in equilibrium. Incentive-compatibility captures the essence of designing a mechanism to overcome the self-interest of agents— in an incentive-compatible mechanism an agent will choose to report its private information truthfully, out of its own self-interest.

Example. The second-price sealed-bid (Vickrey) auction is an incentive-compatible (actually strategy-proof) direct-revelation mechanism for the single-item allocation problem.

Computationally, the revelation principle must be viewed with great suspicion. Direct-revelation mechanisms are often too expensive for agents because they place very high demands on information revelation. An iterative mechanism can sometimes implement the same outcome as a direct-revelation mechanism but with less information revelation and agent computation. The revelation principle restricts *what* we can do, but does not explain *how* to construct a mechanism to achieve a particular set of properties. This is discussed further in Chapter 3.

2.3.1 Incentive Compatibility and Strategy-Proofness

In a direct-revelation mechanism the only action available to an agent is to submit a claim about its preferences.

DEFINITION 2.20 [direct-revelation mechanism] A direct-revelation mechanism $\mathcal{M} = (\Theta_1, \ldots, \Theta_I, g(\cdot))$ restricts the strategy set $\Sigma_i = \Theta_i$ for all i, and has outcome rule $g : \Theta_1 \times \ldots \times \Theta_I \rightarrow \mathcal{O}$ which selects an outcome $g(\hat{\theta})$ based on reported preferences $\hat{\theta} = (\hat{\theta}_1, \ldots, \hat{\theta}_I)$.

In other words, in a direct-revelation mechanism the strategy of agent i is to report type $\hat{\theta}_i = s_i(\theta_i)$, based on its actual preferences θ_i.

A *truth-revealing* strategy is to report true information about preferences, for all possible preferences:

DEFINITION 2.21 [truth-revelation] A strategy $s_i \in \Sigma_i$ is truth-revealing if $s_i(\theta_i) = \theta_i$ for all $\theta_i \in \Theta_i$.

In an *incentive-compatible* (IC) mechanism the equilibrium strategy profile $s^* = (s_1^*,$

$\ldots, s_I^*)$ has every agent reporting its true preferences to the mechanism. We first define Bayesian-Nash incentive-compatibility:

DEFINITION 2.22 [Bayesian-Nash incentive compatible] A direct-revelation mechanism $\mathcal{M}$ is Bayesian-Nash incentive-compatible if truth-revelation is a Bayesian-Nash equilibrium of the game induced by the mechanism.

In other words, in an incentive-compatible mechanism every agent's expected utility maximizing strategy in equilibrium with every other agent is to report its true preferences.

A mechanism is *strategy-proof* (or dominant-strategy incentive-compatible) if truth-revelation is a *dominant-strategy* equilibrium:

DEFINITION 2.23 [strategy-proof] A direct-revelation mechanism $\mathcal{M}$ is strategy-proof if it truth-revelation is a dominant-strategy equilibrium.

Strategy-proofness is a very useful property, both game-theoretically and computationally. Dominant-strategy implementation is very robust to assumptions about agents, such as the information and rationality of agents. Computationally, an agent can compute its optimal strategy without modeling the preferences and strategies of other agents.

A simple equivalence exists between the outcome function $g(\hat{\theta})$ in a direct-revelation mechanism, which selects an outcome based on reported types $\hat{\theta}$ and the social choice function $f(\theta)$ *implemented* by the mechanism, i.e. computed in equilibrium.

PROPOSITION 2.1 (incentive-compatible implementation). *An incentive-compatible direct-revelation mechanism* $\mathcal{M}$ implements *social choice function* $f(\theta) = g(\theta)$, *where* $g(\theta)$ *is the* outcome rule *of the mechanism.*

In other words, in an incentive-compatible mechanism the outcome rule is precisely the social choice function implemented by the mechanism. In Section 2.4 we introduce the *Groves* mechanisms, which are strategy-proof efficient mechanisms for agents with quasi-linear preferences, i.e. the choice rule $k(\hat{\theta})$ computes the efficient allocation given reported types $\hat{\theta}$ and an agent's dominant strategy is truth-revelation.

2.3.2 The Revelation Principle

The *revelation principle* states that under quite weak conditions any mechanism can be transformed into an equivalent *incentive-compatible direct-revelation mechanism* that implements the same social-choice function. The revelation principle is an important tool for the theoretical analysis of what is possible, and of what is impossible, in mechanism design. The revelation principle was first formulated for dominant-strategy equilibria [Gib73], and later extended by Green & Laffont [GJJ77] and Myerson [Mye79, Mye81].

One interpretation of the revelation principle is that incentive-compatibility comes for free. This is not to say that truth-revelation is easy to achieve, but simply that if an indirect-revelation and/or non-truthful mechanism solves a distributed optimization problem, then we would also expect a direct-revelation truthful implementation.

The revelation principle for dominant strategy implementation states that any social choice function than is implementable in dominant strategy is also implementable in a strategy-proof mechanism. In other words it is possible to restrict attention to truth-revealing direct-revelation mechanisms.

THEOREM 2.1 (Revelation Principle). *Suppose there exists a mechanism (direct or otherwise) $\mathcal{M}$ that implements the social-choice function $f(\cdot)$ in dominant strategies. Then $f(\cdot)$ is* truthfully implementable *in dominant strategies, i.e. in a strategy-proof mechanism.*

PROOF. If $\mathcal{M} = (\Sigma_1, \ldots, \Sigma_I, g(\cdot))$ implements $f(\cdot)$ in dominant strategies, then there exists a profile of strategies $s^*(\cdot) = (s_1^*(\cdot), \ldots, s_I^*(\cdot))$ such that $g(s^*(\theta)) = f(\theta)$ for all θ, and for all i and all $\theta_i \in \Theta_i$,

$$u_i(g(s_i^*(\theta_i), s_{-i}), \theta_i) \geq u_i(g(\hat{s}_i, s_{-i}), \theta_i)$$

for all $\hat{s}_i \in \Sigma_i$ and all $s_{-i} \in \Sigma_{-i}$, by definition of dominant strategy implementation. Substituting $s_{-i}^*(\theta_{-i})$ for s_{-i} and $s_i^*(\hat{\theta}_i)$ for $\hat{s}_i$, we have:

$$u_i(g(s_i^*(\theta_i), s_{-i}^*(\theta_{-i})), \theta_i) \geq u_i(g(s_i^*(\hat{\theta}_i, s_{-i}^*(\theta_{-i})), \theta_i)$$

for all $\hat{\theta}_i \in \Theta_i$ and all $\theta_{-i} \in \Theta_{-i}$. Finally, since $g(s^*(\theta)) = f(\theta)$ for all θ, we have:

$$u_i(f(\theta_i, \theta_{-i}), \theta_i) \geq u_i(f(\hat{\theta}_i, \theta_{-i}), \theta_i)$$

for all $\hat{\theta}_i \in \Theta_i$ and all $\theta_{-i} \in \Theta_{-i}$. This is precisely the condition required for $f(\cdot)$ to be truthfully implementable in dominant strategies in a direct-revelation mechanism. The outcome rule in the strategy-proof mechanism, $g : \theta_1 \times \ldots \times \theta_I \rightarrow \mathcal{O}$, is simply equal to the social choice function $f(\cdot)$. ∎

The intuition behind the revelation principle is as follows. Suppose that it is possible to *simulate* the entire system— the bidding strategies of agents and the outcome rule — of an indirect mechanism, given complete and perfect information about the preferences of every agent. Now, if it is possible to claim credibly that the "simulator" will implement an agent's optimal strategy faithfully, given information about the preferences (or type) of the agent, then it is optimal for an agent to truthfully report its preferences to the new mechanism.

This dominant-strategy revelation principle is quite striking. In particular, it suggests that to identify which social choice functions are implementable in dominant strategies, we need only identify those functions $f(\cdot)$ for which truth-revelation is a dominant strategy for agents in a direct-revelation mechanism with outcome rule $g(\cdot) = f(\cdot)$.

A similar revelation principle can be stated in Bayesian-Nash equilibrium.

THEOREM 2.2 (Bayesian-Nash Revelation Principle). *Suppose there exists a mechanism (direct or otherwise) $\mathcal{M}$ that implements the social-choice function $f(\cdot)$ in Bayesian-Nash equilibrium. Then $f(\cdot)$ is truthfully implementable in a (Bayesian-Nash) incentive-compatible direct-revelation mechanism.*

In other words, if the goal is to implement a particular social choice function in Bayesian-Nash equilibrium, it is sufficient to consider only incentive-compatible direct-revelation mechanisms.

The proof closely follows that of the dominant-strategy revelation principle. One problem with the revelation principle for Bayesian-Nash implementation is that the distribution over agent types must be common knowledge to the direct-revelation mechanism, in addition to the agents.

2.3.3 Implications

With the revelation principle in hand we can prove *impossibility results* over the space of direct-revelation mechanisms, and construct *possibility results* over the space of direct-revelation mechanisms.

However, the revelation principle ignores computational considerations and should *not* be taken as a statement that it is sufficient to consider only direct-revelation mechanisms in practical mechanism design. The revelation principle states what can be achieved, what cannot be achieved, but without stating the *computational structure* to achieve a particular set of properties. In particular, in my dissertation I argue that iterative and indirect mechanisms are important in many combinatorial applications, and can provide tractable solutions to problems in which single-shot direct-revelation mechanisms fail.

Rather, the revelation principle provides a rich structure to the mechanism design problem, focusing goals and delineating what *is* and *is not* possible. For example, if a particular direct-revelation mechanism $\mathcal{M}$ is the only mechanism with a particular combination of properties, *then any mechanism, including iterative and indirect mechanisms*, must implement the same outcome (e.g. allocation and payments) as mechanism $\mathcal{M}$ for the same agent preferences.

For example:

- Suppose that the only direct mechanisms with useful properties P1, P2 and P3 are in the class of mechanisms $\mathcal{M}'$. It follows that any mechanism m with properties P1, P2 and P3 must be "outcome equivalent" to a direct mechanism in $\mathcal{M}'$, in the sense that m must implement the same outcome as a mechanism in this class for all possible agent types.

- Suppose that *no* direct mechanism has properties P1, P2 and P3. It follows that there can be no mechanism (direct or otherwise) with properties P1, P2 and P3.

The next section introduces an important family of mechanisms with dominant-strategy solutions.

2.4 Vickrey-Clarke-Groves Mechanisms

In seminal papers, Vickrey [Vic61], Clarke [Cla71] and Groves [Gro73], proposed the Vickrey-Clarke-Groves family of mechanisms, often simply called the Groves mechanisms, for problems in which agents have quasi-linear preferences. The Groves mechanisms are allocatively-efficient and strategy-proof direct-revelation mechanisms.

In special cases there is a Groves mechanism that is also individual-rational and satisfies *weak budget-balance*, such that the mechanism does not require an outside subsidy to operate. This is the case, for example, in the Vickrey-Clarke-Groves mechanism for a combinatorial auction.

In fact, the Groves family of mechanisms characterize the *only* mechanisms that are allocatively-efficient and strategy-proof [GJJ77] amongst direct-revelation mechanisms.

THEOREM 2.3 (Groves Uniqueness). *The Groves mechanisms are the only allocatively-efficient and strategy-proof mechanisms for agents with quasi-linear preferences and general valuation functions, amongst all direct-revelation mechanisms.*

The revelation principle extends this uniqueness to general mechanisms, direct or otherwise. Given the premise that iterative mechanisms often have preferable computational properties in comparison to sealed-bid mechanisms, this uniquenss suggests a focus on *iterative Groves mechanisms* because:

any iterative mechanism that achieves allocative efficiency in dominant-strategy implementation must implement a Groves outcome.

In fact, we will see in Chapter 7 that an iterative mechanism that implements the Vickrey outcome can have slightly weaker properties than those of a single-shot Vickrey scheme.

Krishna & Perry [KP98] and Williams [Wil99] have recently proved the uniqueness of Groves mechanisms among efficient and Bayesian-Nash mechanisms.

2.4.1 The Groves Mechanism

Consider a set of possible alternatives, $\mathcal{K}$, and agents with quasi-linear utility functions, such that

$$u_i(k, p_i, \theta_i) = v_i(k, \theta_i) \Leftrightarrow p_i$$

where $v_i(k, \theta_i)$ is the agent's *value* for alternative k, and p_i is a payment by the agent to the mechanism. Recall that the *type* $\theta_i \in \Theta_i$ is a convenient way to express the valuation function of an agent.

In a direct-revelation mechanism for quasi-linear preferences we write the outcome rule $g(\hat{\theta})$ in terms of a *choice rule*, $k : \Theta_1 \times \ldots \times \Theta_I \rightarrow \mathcal{K}$, and a *payment rule*, $t_i : \Theta_1 \times \ldots \times \Theta_I \rightarrow \mathbb{R}$, for each agent i.

In a Groves mechanism agent i reports type $\hat{\theta}_i = s_i(\theta_i)$, which may not be its true type. Given reported types $\hat{\theta} = (\hat{\theta_1}, \ldots, \hat{\theta_I})$, the choice rule in a Groves mechanism computes:

$$k^*(\hat{\theta}) = \arg\max_{k \in \mathcal{K}} \sum_i v_i(k, \hat{\theta}_i) \tag{1}$$

Choice k^* is the selection that maximizes the total reported value over all agents.

The payment rule in a Groves mechanism is defined as:

$$t_i(\hat{\theta}) = h_i(\hat{\theta}_{-i}) \Leftrightarrow \sum_{j \neq i} v_j(k^*, \hat{\theta}_j) \tag{2.1}$$

where $h_i : \Theta_{-i} \rightarrow \mathbb{R}$ is an arbitrary function on the reported types of every agent except i. This freedom in selecting $h_i(\cdot)$ leads to the description of a "family" of mechanisms. Different choices make different tradeoffs across budget-balance and individual-rationality.

2.4.2 Analysis

Groves mechanisms are efficient and strategy-proof:

THEOREM 2.4 (Groves mechanisms). *Groves mechanisms are allocatively-efficient and strategy-proof for agents with quasi-linear preferences.*

PROOF.

We prove that Groves mechanisms are strategy-proof, such that truth-revelation is a dominant strategy for each agent, from which allocative efficiency follows immediately because the choice rule $k^*(\theta)$ computes the efficient allocation (1).

The utility to agent i from strategy $\hat{\theta}_i$ is:

$$\begin{aligned} u_i(\hat{\theta}_i) &= v_i(k^*(\hat{\theta}), \theta_i) \Leftrightarrow t_i(\hat{\theta}) \\ &= v_i(k^*(\hat{\theta}), \theta_i) + \sum_{j \neq i} v_j(k^*(\hat{\theta}), \hat{\theta}_j) \Leftrightarrow h_i(\hat{\theta}_{-i}) \end{aligned}$$

Ignoring the final term, because $h_i(\hat{\theta}_{-i})$ is independent of an agent i's reported type, we prove that truth-revelation $\hat{\theta}_i = \theta_i$ solves:

$$\max_{\hat{\theta}_i \in \Theta_i} \left[v_i(k^*(\hat{\theta}_i, \hat{\theta}_{-i}), \theta_i) + \sum_{j \neq i} v_j(k^*(\hat{\theta}_i, \hat{\theta}_{-i}), \hat{\theta}_j) \right]$$

$$= \max_{\hat{\theta}_i \in \Theta_i} \left[v_i(x, \theta_i) + \sum_{j \neq i} v_j(x, \hat{\theta}_j) \right] \quad (2)$$

where $x = k^*(\hat{\theta}_i, \hat{\theta}_{-i})$ is the outcome selected by the mechanism. The only effect of the agent's announced type $\hat{\theta}_i$ is on x, and the agent can maximize (2) by announcing $\hat{\theta}_i = \theta_i$ because then the mechanism computes $k^*(\hat{\theta}_i, \hat{\theta}_{-i})$ to explicitly solve:

$$\max_{k \in \mathcal{K}} v_i(k, \theta_i) + \sum_{j \neq i} v_j(k, \hat{\theta}_j)$$

Truth-revelation is the *dominant strategy* of agent i, whatever the reported types $\hat{\theta}_{-i}$ of the other agents. ▌

The effect of payment $t_i(\hat{\theta}) = (\cdot) \Leftrightarrow \sum_{j \neq i} v_j(k^*, \hat{\theta}_j)$ is to "internalize the externality" placed on the other agents in the system by the reported preferences of agent i. This aligns the agents' incentives with the system-wide goal of an efficient allocation, an agent *wants* the mechanism to select the best system-wide solution given the reports of other agents and its own *true* preferences.

The first term in the payment rule, $h_i(\hat{\theta}_{-i})$, can be used to achieve (weak) budget-balance and/or individual rationality. It is not possible to simply total up the payments made to each agent in the Groves scheme and divide equally across agents, because the total payments depend on the outcome, and therefore the reported type of each agent. This would break the strategy-proofness of the mechanism.

2.4.3 The Vickrey Auction

The special case of Clarke mechanism for the allocation of a single item is the familiar second-price sealed-bid auction, or Vickrey [Vic61] auction.

In this case, with bids b_1 and b_2 to indicate the first- and second- highest bids, the item is sold to the item with the highest bid (agent 1), for a price computed as:

$$b_1 \Leftrightarrow (b_1 \Leftrightarrow b_2) = b_2$$

i.e. the *second-highest* bid.

One can get some intuition for the strategy-proofness of the Groves mechanisms in this special case. Truth-revelation is a dominant strategy in the Vickrey auction because an agent's bid determines the *range* of prices that it will accept, but not the actual price it pays. The price that an agent pays is completely independent of its bid price, and even if an agent knows the second-highest bid it can still bid its true value because it only pays just enough to out-bid the other agent. In addition, notice that *weak* budget-balance holds, because the second-highest bid price is non-negative, and *individual-rationality* holds because the second-highest bid price is no greater than the highest bid price, which is equal to the winner agent's value in equilibrium.

2.4.4 The Pivotal Mechanism

The Pivotal, or Clarke, mechanism [Cla71] is a Groves mechanism in which the payment rule, $h_i(\hat{\theta}_{-i})$, is carefully set to achieve individual-rationality, while also maximizing the payments made by the agents to the mechanism. The Pivotal mechanism also achieves *weak* budget-balance whenever that is possible in an efficient and strategy-proof mechanism [KP98].

The Clarke mechanism [Cla71] computes the additional transfer term as:

$$h_i(\hat{\theta}_{-i}) = \sum_{j \neq i} v_j(k^*_{-i}(\hat{\theta}_{-i}), \hat{\theta}_j) \tag{2.2}$$

where $k^*_{-i}(\hat{\theta}_{-i})$ is the *optimal collective choice* for with agent i taken out of the system:

$$k^*_{-i}(\hat{\theta}_{-i}) = \arg\max_{k \in \mathcal{K}} \sum_{j \neq i} v_j(k, \hat{\theta}_j)$$

This is a valid additional transfer term because the reported value of the second-best allocation without agent i is *independent* of the report from agent i. The strategy-proofness and efficiency of the Groves mechanisms are left unchanged.

The Clarke mechanism is a useful special-case because it is also *individual rational* in quite general settings, which means that agents will choose to participate in the mechanism (see Section 2.2.2).

To keep things simple, let us assume that agent i's expected utility from not participating in the mechanism is $\overline{u}_i(\theta_i) = 0$. The Clarke mechanism is individual rational when the following two (sufficient) conditions hold on agent preferences:

DEFINITION 2.24 [choice set monotonicity] The feasible choice set available to the mechanism $\mathcal{K}$ (weakly) increases as additional agents are introduced into the system.

DEFINITION 2.25 [no negative externalities] Agent i has non-negative value, i.e. $v_i(k^*_{-i}, \theta_i) \geq 0$, for any optimal solution choice, $k^*_{-i}(\theta_{-i})$ without agent i, for all i and all θ_i.

In other words, with *choice set monotonicity* an agent cannot "block" a selection, and with *no negative externalities*, then any choice not involving an agent has a neutral (or positive) effect on that agent.

For example, the conditions of choice-set monotonicity and no negative externalities hold in the following settings:

- In a *private goods* market environment: introducing a new agent cannot make existing trades infeasible (in fact it can only increase the range of possible trades); and with only private goods no agent has a negative value for the trades executed between other agents (relative to no trades).

- In a *public project* choice problem: introducing a new agent cannot change the range of public projects that can be implemented; and no agent has negative value for any public project (relative to the project not going ahead).

PROPOSITION 2.2 (Clarke mechanism). *The Pivotal (or Clarke) mechanism is (*ex post*) individual-rational, efficient, and strategy-proof when choice-set monotonicity and no negative externalities hold and with quasi-linear agent preferences.*

PROOF. To show individual-rationality (actually *ex post* individual-rationality), we show that the utility to agent i in the equilibrium outcome of the mechanism is always non-negative. We can assume truth-revelation in equilibrium. The utility to agent i with type θ_i is:

$$\begin{aligned} u_i(\theta_i, \theta_{-i}) &= v_i(k^*(\theta), \theta_i) \Leftrightarrow \left(\sum_{j \neq i} v_j(k^*_{-i}(\theta_{-i}), \theta_j) \Leftrightarrow \sum_{j \neq i} v_j(k^*(\theta), \theta_j) \right) \\ &= \sum_i v_i(k^*(\theta), \theta_i) \Leftrightarrow \sum_{j \neq i} v_j(k^*_{-i}(\theta_{-i}), \theta_j) \end{aligned} \tag{3}$$

Expression (3) is non-negative because the value of the best solution without agent i, $\sum_{j \neq i} v_j(k^*_{-i}(\theta_{-i}), \theta_j)$, cannot be greater than the value of the best solution with agent i, $\sum_i v_i(k^*(\theta), \theta_i)$. This follows because any choice with agents $j \neq i$ is also feasible with all agents (*monotonicity*), and has just as much total value (*no negative externalities*). ∎

The Clarke mechanism also achieves *weak* budget-balance in special-cases. A sufficient condition is the *no single-agent effect*:

DEFINITION 2.26 [no single-agent effect] For any collective choice k' that is optimal in some scenario with all agents, i.e. $k' = \max_{k \in \mathcal{K}} \sum_i v_i(k, \theta_i)$, for some $\theta \in \Theta$, then for all i there must exist another choice k_{-i} that is feasible without i and has as much value to the remaining agents $j \neq i$.

In words, the *no single-agent effect* condition states that any one agent can be removed from an optimal system-wide solution without having a negative effect on the best choice available to the remaining agents. This condition holds in the following settings:

- In an auction with only buyers (i.e. the auctioneer holds all the items for sale), so long as all buyers have "free disposal", such that they have at least as much value for more items than less items.
- In a public project choice, because the set of choices available is static, however many agents are in the system.

PROPOSITION 2.3 (Clarke weak budget-balance). *The Pivotal (or Clarke) mechanism is (ex post) individual-rational, weak budget-balanced, efficient and strategy-proof when choice-set monotonicity, no negative externalities, and no single-agent effect hold, and with quasi-linear agent preferences.*

PROOF. Again, we can assume truth-revelation in equilibrium, and prove that the total transfers are non-negative, such that the mechanism does not require a subsidy, i.e.

$$\sum_i t_i(\theta) \geq 0$$

for all $\theta \in \Theta$. Substituting the expression for agent transfers, we have:

$$\sum_i \left(\sum_{j \neq i} v_j(k^*_{-i}(\theta_{-i}), \theta_j) \Leftrightarrow \sum_{j \neq i} v_j(k^*(\theta), \theta_j) \right) \geq 0$$

This is satisfied in Clarke because the transfer is non-negative for *every* agent i, i.e.:

$$\sum_{j \neq i} v_j(k^*_{-i}(\theta_{-i}), \theta_j) \geq \sum_{j \neq i} v_j(k^*(\theta), \theta_j), \qquad \forall i$$

This condition holds by a simple feasibility argument with the no single-agent effect, because any solution to the system with all agents remains feasible and has positive value without any one agent. ∎

As soon as there are buyers and sellers in a market we very quickly lose even weak budget-balance with Groves-Clarke mechanisms. The budget-balance problem in a combinatorial exchange is addressed in Parkes, Kalagnanam & Eso [PKE01], where we propose a number of methods to trade-off strategy-proofness and allocative efficiency for budget-balance.

2.4.5 The Generalized Vickrey Auction

The Generalized Vickrey Auction is an application of the Pivotal mechanism to the combinatorial allocation problem. The combinatorial allocation problem (CAP) was introduced in Section 1.2. There are a set $\mathcal{G}$ of items to allocate to $\mathcal{I}$ agents. The set of choices $\mathcal{K} = \{(S_1, \ldots, S_I) \ : \ S_i \cap S_j = \emptyset, S_i \subseteq \mathcal{G}\}$ where S_i is an allocation of a bundle of items to agent i. Given preferences (or type) θ_i, each agent i has a quasi-linear utility function, $u_i(S, p_i, \theta_i) = v_i(S, \theta_i) \Leftrightarrow p_i$, for bundle S and payment p_i. For notational simplicity we will drop the "type" notation in this section, and simply write $v_i(S, \theta_i) = v_i(S)$.

The efficient allocation computes an allocation to maximize the total value:

$$S^* = \arg \max_{S=(S_1,\ldots,S_I)} \sum_i v_i(S_i)$$
$$\text{s.t.} \quad S_i \cap S_j = \emptyset, \qquad \text{for all } i, j$$

The Pivotal mechanism applied to this problem is a sealed-bid combinatorial auction, often called the *Generalized Vickrey Auction* (GVA). The special case for a single item is the Vickrey auction. In the GVA each agent bids a value for all possible sets of items, and the mechanism computes an allocation and payments.

The GVA has the following useful properties:

THEOREM 2.5 (Generalized Vickrey Auction). *The GVA is efficient, strategy-proof, individual-rational, and weak budget-balanced for agents with quasi-linear preferences in the combinatorial allocation problem.*

Description

Each agent $i \in \mathcal{I}$ submits a (possibly untruthful) valuation function, $\hat{v}_i(S)$, to the auctioneer. The outcome rule in the Pivotal mechanism computes $k^*(\hat{\theta})$, the allocation that maximizes reported value over all agents. In the GVA this is equivalent to the auctioneer solving a "winner-determination" problem, solving CAP with the reported values and computing allocation $\mathbf{S}^* = (S_1^*, \ldots, S_I^*)$ to maximize reported value. Let V^* denote the total value of this allocation. Allocation $\mathbf{S}^*$ is the allocation implemented by the auctioneer.

The payment rule in the Pivotal mechanism also requires that the auctioneer solves a smaller CAP, with each agent i taken out in turn, to compute $k^*_{-i}(\theta_{-i})$, the best allocation without agent i. Let $(S_{-i})^*$ denote this second-best allocation, and $(V_{-i})^*$ denote its value.

Finally, from the Groves-Clarke payment rule $t_i(\hat{\theta})$, see (2.1) and (2.2), the auctioneer computes agent i's payment as:

$$p_{\text{vick}}(i) = (V_{-i})^* \Leftrightarrow \sum_{j \neq i} \hat{v}_j(S_j^*)$$

In words, an agent pays the marginal negative effect that its participation has on the (reported) value of the other agents. Equivalently, the Vickrey payment can be formulated as a discount $\Delta_{\text{vick}}(i)$ from its bid price, $\hat{v}_i(S_i^*)$, i.e. $p_{\text{vick}}(i) = \hat{v}_i(S_i^*) \Leftrightarrow \Delta_{\text{vick}}(i)$, for *Vickrey discount*:

$$\Delta_{\text{vick}}(i) = V^* \Leftrightarrow (V_{-i})^*$$

Analysis

Efficiency and strategy-proofness follow immediately from the properties of the Groves mechanism. Weak budget-balance also holds; it is simple to show that each agent pays a non-negative amount to the auctioneer by a simple feasibility argument. Individual-rationality also holds, and agents pay no more than their value for the bundle they receive;

Name	Preferences	Solution concept	Impossible	Environment
GibSat	general	dominant	Non-dictatorial (incl. Pareto Optimal)	general
Hurwicz	quasi-linear	dominant	Eff& BB	simple-exchange
MyerSat	quasi-linear	Bayesian-Nash	Eff& BB & IR	simple-exchange
GrLaff	quasi-linear	coalition-proof	Eff	simple-exchange

Table 2.1: Mechanism design: Impossibility results. *Eff* is *ex post* allocative efficiency, *BB* is *ex post* (and strong) budget-balance, and *IR* is *interim* individual rationality.

it is simple to show that discounts are always non-negative, again by a simple feasibility argument. Alternatively, one can verify that conditions choice-set monotonicity, no negative externalities, and no single-agent effect hold for the CAP.

2.5 Impossibility Results

The revelation principle allows the derivation of a number of impossibility theorems that outline the combinations of properties that *no* mechanism can achieve (with fully rational agents) in particular types of environments. The basic approach to show impossibility is to assume direct-revelation and incentive-compatibility, express the desired properties of an outcome rule as a set of mathematical conditions (including conditions for incentive-compatibility), and then show a conflict across the conditions.

Table 2.1 describes the main impossibility results. Results are delineated by *conditions on agent preferences*, the *equilibrium solution concept*, and the assumptions about the *environment*. The "Impossible" column lists the combinations of desirable mechanism properties that cannot be achieved in each case.

As discussed in Section 2.2.2, *ex post* refers to conditions tested at the outcome of the mechanism. *Interim* individual-rationality means that an agent that knows its own preferences but only has distributional information about the preferences of other agents will choose to participate in the mechanism.

A few words about the interpretation of impossibility results are probably useful. Impossibility for restricted preferences in an exchange is more severe than for general preferences and general environments, because general conditions include these as special cases.

In addition, impossibility for weak solution concepts such as Bayesian-Nash is more restrictive than impossibility for strong solution concepts like dominant strategy implementation.

We also need a few more definitions:

DEFINITION 2.27 [dictatorial] A social-choice function is *dictatorial* if one (or more) agents always receives one of its most-preferred alternatives.

DEFINITION 2.28 [general preferences] Preferences θ_i are *general* when they provide a complete and transitive preference ordering $\succ$ on outcomes. An ordering is *complete* if for all $o_1, o_2 \in \mathcal{O}$, we have $o_1 \succ o_2$ or $o_2 \succ o_1$ (or both). An ordering is *transitive* if for all $o_1, o_2, o_3 \in \mathcal{O}$, if $o_1 \succ o_2$ and $o_2 \succ o_3$ then $o_1 \succ o_3$.

DEFINITION 2.29 [coalition-proof] A mechanism $\mathcal{M}$ is *coalition-proof* if truth revelation is a dominant strategy for any coalition of agents, where a coalition is able to make side-payments and re-distribute items after the mechanism terminates.

DEFINITION 2.30 [general environment] A *general* environment is one in which there is a discrete set of possible outcomes $\mathcal{O}$ and agents have general preferences.

DEFINITION 2.31 [simple exchange] A *simple exchange* environment is one in which there are buyers and sellers, selling single units of the same good.

The Gibbard [Gib73] and Satterthwaite [Sat75] impossibility theorem shows that for a *sufficiently rich class* of agent preferences it is impossible to implement a satisfactory social choice function in dominant strategies. A related impossibility result, due to Green and Laffont [GJJ77] and Hurwicz [Hur75], demonstrates the impossibility of efficiency and budget-balance with dominant strategy implementation, even in quasi-linear environments.

More recently, the Myerson-Satterthwaite impossibility theorem [Mye83] extends this impossibility to include Bayesian-Nash implementation, if *interim* individual-rationality is also required. Williams [Wil99] and Krishna & Perry [KP98] provide alternative derivations of this general impossibility theorem, using properties about the Groves family of mechanisms.

Green & Laffont [GL79] demonstrate that no allocatively-efficient and strategy-proof mechanism can also be safe from manipulation by coalitions, even in quasi-linear environments.

The following sections describe the results in more details.

2.5.1 Gibbard-Satterthwaite Impossibility Theorem

A negative result due to Gibbard [Gib73] and Satterthwaite [Sat75] states that it is impossible, in a sufficiently rich environment, to implement a non-dictatorial social-choice function in dominant strategy equilibrium.

THEOREM 2.6 (Gibbard-Satterthwaite Impossibility Theorem). *If agents have general preferences, and there are at least two agents, and at least three different optimal outcomes over the set of all agent preferences, then a social-choice function is dominant-strategy implementable if and only if it is dictatorial.*

Clearly all dictatorial social-choice functions must be strategy-proof. This is simple to show because the outcome that is selected is the most preferred, or maximal outcome, for the reported preferences of one (or more) of the agents— so an agent should report its true preferences. For a proof in the other direction, that any strategy-proof social-choice function must be dictatorial, see MasColell *et al.* [MCWG95].

Impossibility results such as Gibbard-Satterthwaite must be interpreted with great care. In particular the results do not necessarily continue to hold in *restricted environments*. For example, although no dictatorial social choice function can be Pareto optimal or efficient, this impossibility result does not apply directly to markets. The market environment naturally imposes additional structure on preferences. In particular, the Gibbard-Satterthwaite impossibility theorem may not hold if one of the following conditions are relaxed:

— additional constraints on agent preferences (e.g. quasi-linear)

— weaker implementation concepts, e.g. Bayesian-Nash implementation

In fact a market environment has been shown to make implementation easier. Section 2.6 introduces a number of *non-dictatorial* and *strategy-proof* mechanisms in restricted environments; e.g. McAfee [McA92] for quasi-linear preferences in a double-auction, and Barberà & Jackson [BJ95] for quasi-concave preferences in a classic exchange economy.

2.5.2 Hurwicz Impossibility Theorem

The Hurwicz impossibility theorem [Hur75] is applicable to even *simple* exchange economies, and for agents with quasi-linear preferences. It states that it is impossible to implement

an efficient and budget-balanced social choice function in dominant-strategy in market settings, even without requiring individual-rationality and even with additional restrictions on agent valuation functions.[1]

Hurwicz [Hur72] first showed a conflict between efficiency and strategy-proofness in a simple two agent model. The general impossibility result follows from Green & Laffont [GJJ77] and Hurwicz [Hur75], and more recently Hurwicz and Walker [HW90]. Green & Laffont and Hurwicz established that no member of the Groves family of mechanisms has budget-balance, and that the Groves family is the unique set of strategy-proof implementation rules in a simple exchange economy. I find it useful to refer to this result as the Hurwicz impossibility theorem.

THEOREM 2.7 (Hurwicz Impossibility Theorem). *It is impossible to implement an efficient, budget-balanced, and strategy-proof mechanism in a simple exchange economy with quasi-linear preferences.*

This result is quite negative, and suggests that if allocative efficiency and budget-balance are required in a simple exchange economy, then looking for dominant strategy solutions is not useful (via the revelation principle). Fortunately, strong budget-balance if often not necessary, and we can achieve strategy-proofness, efficiency and weak budget-balance via the Vickrey-Clarke-Groves mechanisms in a number of interesting domains.

2.5.3 Myerson-Satterthwaite Impossibility Theorem

The Myerson-Satterthwaite impossibility theorem [Mye83] strengthens the Hurwicz impossibility result to include Bayesian-Nash implementation, if *interim* individual-rationality is also required.

THEOREM 2.8 (Myerson-Satterthwaite). *It is impossible to achieve allocative efficiency, budget-balance and* (interim) *individual-rationality in a Bayesian-Nash incentive-compatible mechanism, even with quasi-linear utility functions.*

[1] Schummer [Sch97] has recently shown that even for the case of two agents with linear preferences it is not possible to achieve strategy-proofness and efficiency.

Name	Pref	Solution	Possible	Environment	
Groves	quasi-linear	dominant	Eff & (IR *or* WBB)	exchange	VCG
dAGVA	quasi-linear	Bayesian-Nash	Eff & BB	exchange	[dG79, Arr79]
Clarke	quasi-linear	dominant	Eff & IR	exchange	[Cla71]
GVA	quasi-linear	dominant	Eff, IR & WBB	comb auction	VCG
MDP	classic	iterative local-Nash	Pareto	exchange	[DdlVP71, Mal72 Rob79]
BJ95	classic	dominant	BB & non-dictatorial	exchange	[BJ95]
Quadratic	classic	Nash	Pareto & IR	exchange	[GL77b]

Table 2.2: Mechanism design: Possibility results. *Eff* is *ex post* allocative efficiency, *BB* is *ex post* strong budget-balance, *WBB* is *ex post* weak budget-balance, *IR* is *interim* individual-rationality, *Pareto* is *ex post* Pareto-optimality.

Myerson & Satterthwaite [Mye83] demonstrate this impossibility in a two-agent one-good example, for the case that trade is possible but not certain (e.g. the buyer and seller have overlapping valuation ranges). Williams [Wil99] and Krishna & Perry [KP98] provide alternative derivations of this general impossibility result, using properties of the Groves family of mechanisms.

An immediate consequence of this result is that we can only hope to achieve at most *two* of Eff, IR and BB in an market with quasi-linear agent preferences, even if we look for Bayesian-Nash implementation. The interested reader can consult Laffont & Maskin [LM82] for a technical discussion of various approaches to achieve any two of these three properties.

In the next section we introduce the dAGVA mechanism [Arr79, dG79], that is able to achieve efficiency and budget-balance, but loses individual-rationality. The dAGVA mechanism is an "expected Groves mechanism."

2.6 Possibility Results

The central positive result is the family of Vickrey-Clarke-Groves (VCG) mechanisms, which are allocatively-efficient (but not budget-balanced) strategy-proof mechanisms in quasi-linear domains. VCG mechanisms clearly demonstrate that it is possible to implement non-dictatorial social choice functions in more restricted domains of preferences. However, as expected from the impossibility results of Green & Laffont [GJJ77] and Hurwicz [Hur75], they are not efficient *and strong* budget-balanced.

Table 2.2 summarizes the most important possibility results. A quick check confirms

that these possibility results are all consistent with the impossibility results of Table 2.1! By the revelation principle we effectively get "incentive-compatibility" for free in direct-revelation mechanisms, and these are all incentive-compatible except the iterative MDP procedure.

The possibility results are delineated by *agent preferences*, the *equilibrium solution concept* and the *environment* or problem domain.

We need a few additional definitions to explain the characterization.

DEFINITION 2.32 [classic preferences] Classic preferences are strictly quasi-concave, continuous and increasing utility functions.

DEFINITION 2.33 [exchange environment] Exchange simply refers to a bilateral trading situation, with agents that have general valuation functions (including bundle values).

Contrary to impossibility results, for possibility results a strong implementation concept is more useful than a weak implementation, e.g. dominant is preferred to Bayesian-Nash, and a general environment such as an exchange is preferred to a more restricted environment such as a combinatorial auction (which can be viewed as a one-sided exchange).

The Groves, Clarke (Pivotal), and GVA mechanisms have already been described in Section 2.4. Checking back with the impossibility results: Groves mechanisms are consistent with the Gibbard-Satterthwaite impossibility theorem because agent preferences are not general but quasi-linear;[2] and Groves mechanisms are consistent with the Hurwicz/Myerson-Satterthwaite impossibility theorems because strong budget-balance does not hold. Groves mechanisms are *not* strong budget-balanced. This failure of strong budget-balance can be acceptable in some domains; e.g., in one-sided auctions (combinatorial or otherwise) with a single seller and multiple buyers it may be acceptable to achieve *weak* budget-balance and transfer net payments to the seller.

2.6.1 Efficiency and Strong Budget-Balance: dAGVA

An interesting extension of the Groves mechanism, the *dAGVA* (or "expected Groves") mechanism, due to Arrow [Arr79] and d'Aspremont & Gérard-Varet [dG79], demonstrates that it is possible to achieve efficiency and budget-balance in a Bayesian-Nash equilibrium,

[2] MasColell also notes that there are no dictatorial outcomes in this environment.

even though this is impossible in dominant-strategy equilibrium (Hurwicz). However, the dAGVA mechanism is *not* individual-rational, which we should expect by the Myerson-Satterthwaite impossibility theorem.

THEOREM 2.9 (dAGVA mechanism). *The dAGVA mechanism is* ex ante *individual-rational, Bayesian-Nash incentive-compatible, efficient and (strong) budget-balanced with quasi-linear agent preferences.*

The dAGVA mechanism is a direct-revelation mechanism in which each agent announces a type $\hat{\theta}_i \in \Theta_i$, that need not be its true type. The mechanism is an "expected-form" Groves mechanism [Rob87, KP98].

The allocation rule is the same as for the Groves mechanism:

$$k^*(\hat{\theta}) = \max_{k \in \mathcal{K}} \sum_i v_i(k, \hat{\theta}_i)$$

The structure of the payment rule is also quite similar to that in the Groves mechanism:

$$t_i(\hat{\theta}) = h_i(\hat{\theta}_{-i}) \Leftrightarrow E_{\theta_{-i}} \left[\sum_{j \neq i} v_j(k^*(\hat{\theta}_i, \theta_{-i}), \theta_j) \right]$$

where as before $h(\cdot)$ is an arbitrary function on agents' types. The second term is the expected total value for agents $j \neq i$ when agent i announces type $\hat{\theta}_i$ and agents $j \neq i$ tell the truth. It is a function of only agent i's announcement, not of the actual strategies of agents $j \neq i$, making it a little different from the formulation of agent transfers in the Groves mechanism. In effect, agent i receives a transfer due to this term equal to the *expected* externality of a change in its own reported type on the other agents in the system.

The Bayesian-Nash incentive-compatibility with this transfer follows from a similar line of reasoning as the strategy-proofness of the Groves mechanisms. A proof is in the appendix to this chapter.

The interesting thing about the dAGVA mechanism is that it is possible to choose the $h_i(\cdot)$ functions to satisfy budget-balance, such that $\sum_i t_i(\theta) = 0$ for all $\theta \in \Theta_i$. Define the "expected social welfare (or value)" of agents $j \neq i$ when agent i announces its type θ_i as

$$\mathrm{SW}_{-i}(\hat{\theta}_i) = E_{\theta_{-i}} \left[\sum_{j \neq i} v_j(k^*(\hat{\theta}_i, \theta_{-i}), \theta_j) \right]$$

and note that this does not depend on announced types of agents $j \neq i$. The additional term in the payment rule is defined, for agent i, as:

$$h_i(\hat{\theta}_{-i}) = \left(\frac{1}{I \Leftrightarrow 1}\right) \sum_{j \neq i} \mathrm{SW}_{-j}(\hat{\theta}_j)$$

which is the "averaged" expected social welfare to every other agent given the announced types of agents $j \neq i$. This gives budget-balance because each agent also pays an equal $1/(I \Leftrightarrow 1)$ share of the total payments made to the other agents, none of which depend on its own announced type. See the appendix of this chapter for a proof.

The incentive properties and properties of full optimality, i.e. efficiency and budget-balance, make the dAGVA procedure very attractive. However, the dAGVA mechanism has a number of problems:

(1) it may not satisfy the individual rationality constraint (even *ex ante*)

(2) Bayesian-Nash implementation is much weaker than dominant-strategy implementation

(3) it places high demands on agent information-revelation

Roberts [Rob87] provides a very interesting discussion of the conditions required for an iterative method to implement the dAGVA mechanism with less information from agents. In fact, he claims that it is *impossible* to find a successful iterative procedure because an agent's announcement in earlier periods must also affect its payments in subsequent periods, breaking incentive-compatibility.

2.6.2 Dominant-strategy Budget-Balance with Inefficient Allocations

A number of mechanisms have been proposed to achieve budget-balance (perhaps weak budget-balance) in dominant strategy mechanisms, for some loss in allocative efficiency. McAfee [McA92] presents a mechanism for a double auction (with multiple buyers and sellers) that is strategy-proof and satisfies weak budget-balance, but for some loss in allocative efficiency.

Barberà & Jackson [BJ95] characterize the set of strategy-proof social-choice functions that can be implemented with budget-balance in an exchange economy with classic preferences. Comparing back with the Gibbard-Satterthwaite impossibility theorem [Gib73, Sat75], it is possible to implement non-dictatorial social choice functions in this restricted set of preferences, even though preferences are not quasi-linear. In fact Barberà

& Jackson show that it is necessary and sufficient to implement "fixed-proportion trading rules", in which (loosely speaking) agents trade in pre-specified proportions. Given Hurwicz's impossibility theorem, it is not surprising that the trading rules are not fully allocatively-efficient.

2.6.3 Alternative implementation Concepts

One method to extend the range of social-choice functions that can be implemented is to consider alternative equilibrium solution concepts. In the context of direct-revelation mechanisms (i.e. static games of incomplete information) we have already observed that Bayesian-Nash implementation can help (e.g. in the dAGVA mechanism). One difficulty with Bayesian-Nash implementation is that it requires more information and rationality assumptions of agents. Similarly, we might expect that moving to a Nash implementation concept can help again.

Groves & Ledyard [GL77] inspired much of the literature on Nash implementation. The *Quadratic mechanism* is Pareto efficient in the exchange environment with classic preferences, in that all Nash equilibria are Pareto efficient. In this sense, it is demonstrated that it is possible to implement budget-balanced and efficient outcomes with Nash implementation, while (Myerson-Satterthwaite) this is not possible with Bayesian-Nash.

However, the Nash implementation concept is quite problematic. An agent's Nash strategy depends on the strategies of other agents, and on complete information about the (private) types of each agent. Clearly, it is quite unreasonable to expect agents to select Nash strategies in a one-shot direct-revelation mechanism. The solution concepts only make sense if placed within an iterative procedure, where agents can adjust towards Nash strategies across rounds [Gro79].

Moore & Rupullo [MR88] consider subgame-perfect Nash implementation in dynamic games, and show that this expands the set of social-choice functions that can be implemented in strategy-proof mechanisms. Of course, introducing a new solution concept requires a new justification of the merits of the subgame-perfect refinement to Nash equilibrium in a dynamic game. A fascinating recent idea, due to Kalai & Ledyard [KL98] considers "repeated implementation", in which the authors consider the implementable social-choice functions in a repeated game, with strong results about the effect on implementation.

The mechanism design literature is almost exclusively focused on direct-revelation mechanisms and ignores the costs of information revelation and centralized computation. One exception is the *MDP* planning procedure, proposed by Drèze & de la Vallée Poussin [DdlVP71] and Malinvaud [Mal72]. The MDP mechanism is an iterative procedure, in which in each round each agent announces "gradient" information about its preferences for different outcomes. The center adjusts the outcome towards a Pareto optimal solution in an exchange environment with classic agent preferences. If the agents report truthful information the MDP procedure is Pareto optimal (i.e. fully efficient).

Drèze & de la Vallée Poussin [DdlVP71] also consider the incentives to agents for reporting truthful information in each round, and showed that truthful revelation is a local maximin strategy (i.e. maximizes the utility of an agent given that other agents follow a worst-case strategy). Truth revelation is also a Nash equilibrium at termination.

In addition, Roberts [Rob79] proved that if agents play a local Nash equilibrium at each stage in the procedure, to maximize the immediate increase in utility of the project, then the mechanism will still converge to a Pareto optimum even though the agents do not report truthful information. Roberts retains a myopic assumption, and studied only a local game in which agents did not also consider the effect of information on future rounds.

Champsaur & Laroque [CL82] departed from this assumption of myopic behavior, and assumed that every agent considers the Nash equilibrium over a period of T periods. The agents forecast the strategies of other agents over T periods, and play a Nash equilibrium. The MDP procedure is still Pareto optimal, but the main difference is that the center has much less control over the final outcome (it is less useful as a policy tool). The outcome for large T approaches the competitive equilibrium.

Modeling agents with a Nash equilibrium, even in the local game, still makes the (very) unreasonable assumption that agents have complete information about each others' preferences, for example to compute the equilibrium strategies. Roberts [Rob79] discusses iterative procedures in which truthful revelation locally dominant at each stage. Of course, one must expect some loss of efficiency if strategy-proofness is the goal.

*i*Bundle [Par99, PU00a] is an efficient ascending-price auction for the combinatorial allocation problem, with myopic best-response agent strategies. The auction is weak budget-balanced, and individual-rational. Although myopic best-response is not a rational sequential strategy for an agent, it is certainly a more reasonable implementation concept than

a local Nash strategy, requiring only price information and information about an agent's own preferences. As discussed in Chapter 7, an extended auction, *i*Bundle Extend&Adjust, provably computes VCG payments in many problems. Computing the outcome of a Groves mechanism with myopic best-response strategies makes myopic best-response a Bayesian-Nash equilibrium of the iterative auction.

2.7 Optimal Auction Design

In a seminal paper, Myerson [Mye81] adopted a constructive approach to mechanism design for *private-values* auction, in which an agent's value is independent of that of other agents. Myerson considers an objective of *revenue maximization*, instead of allocative-efficiency, and formulates the mechanism design problem as an optimization problem. The objective is to design an outcome function for a direct-revelation mechanism that maximizes the expected revenue subject to constraints on: *feasibility* (no item can be allocated more than once); *individual-rationality* (the expected utility for participation is non-negative); and *incentive-compatibility.*

Focusing on direct-revelation mechanisms (following the revelation principle), Myerson derives conditions on the allocation rule $k : \Theta \rightarrow \mathcal{K}$ and the payment rules $t_i : \Theta \rightarrow \mathbb{R}$ for an auction to be optimal. Without solving for explicit functional forms $k(\cdot)$ and $t_i(\cdot)$ Myerson is able to derive the *revenue equivalence theorem*, which essentially states that any auction that implements a particular allocation rule $k(\cdot)$ must have the same expected payments.

In general the goals of revenue-maximization and efficiency are in conflict. Myerson constructs an optimal (revenue-maximizing) auction in the simple single-item case, and demonstrates that a seller with distributional information about the values of agents can maximize its expected revenue with an inefficient allocation-rule. The seller announces a non-zero reservation price, which increases its revenue in some cases but also introduces a slight risk that the seller will miss a profitable trade (making it inefficient).

Krishna & Perry [KP98] develop a generalized revenue-equivalence principle:

THEOREM 2.10 (generalized revenue-equivalence). *In quasi-linear environments, all Bayesian-Nash incentive-compatible mechanisms with the same choice rule $k(\cdot)$ are expected revenue equivalent up to an additive constant.*

This is essentially a statement that all mechanisms that implement a particular allocation rule are equivalent in their transfer rules. We have already seen a similar result, i.e. that the Groves mechanisms are unique among efficient & strategy-proof mechanisms [GL87].

Krishna & Perry also show that the GVA maximizes revenue over all efficient and individual-rational mechanisms, even amongst mechanisms with Bayesian-Nash implementation:

THEOREM 2.11 (Revenue-optimality of GVA). *The GVA mechanism maximizes the expected revenue amongst all efficient, (Bayesian-Nash) incentive-compatible, and individual-rational mechanisms.*

It is interesting that the dominant-strategy GVA mechanism maximizes revenue over all Bayesian-Nash incentive-compatible and efficient mechanisms.

Ausubel & Cramton [AC98] make a simpler argument for efficient mechanisms in the presence of after-markets. Intuitively, in the presence of an after-market that will allow agents to achieve an efficient allocation outside of the auction, the auctioneer maximizes profits by providing agents with an allocation that they find most desirable and extracting their surplus. A similar argument can be made in the presence of alternate markets. If the auctioneer does not compute efficient allocations then agents will go elsewhere.

Appendix: Proof of dAGVA properties

The intuition behind the Bayesian-Nash incentive-compatibility of the dAGVA mechanism follows a similar line of reasoning to that for the strategy-proofness of Groves. Suppose that the other agents announce their true types, the expected utility to agent i for announcing its true type (given correct information about the distribution over the types of other

agents) is:

$$E_{\theta_{-i}}\left[v_i(k^*(\theta_i,\theta_{-i}),\theta_i)\right] + E_{\theta_{-i}}\left[\sum_{j\neq i} v_j(k^*(\theta_i,\theta_{-i}),\theta_j)\right]$$

$$= E_{\theta_{-i}}\left[\sum_{j=1}^{I} v_j(k^*(\theta),\theta_j)\right]$$

and this is greater than

$$E_{\theta_{-i}}\left[\sum_{j=1}^{I} v_j(k^*(\hat{\theta}_i,\theta_{-i}),\theta_j)\right]$$

for all $\hat{\theta}_i \in \Theta_i$ because by reporting its true type the agent explicitly instructs the mechanism to compute an allocation that maximizes the inner-term of the expectation for all possible realizations of the types θ_{-i} of the other agents.

Finally, we show that the dAGVA mechanism is budget-balanced:

$$\begin{aligned}
\sum_i t_i(\theta) &= \left(\frac{1}{I \Leftrightarrow 1}\right) \sum_i \sum_{j\neq i} \mathrm{SW}_{-j}(\theta_j) \Leftrightarrow \sum_i \mathrm{SW}_{-i}(\theta_i) \\
&= \left(\frac{1}{I \Leftrightarrow 1}\right) \sum_i (I \Leftrightarrow 1)\mathrm{SW}_{-i}(\theta_i) \Leftrightarrow \sum_i \mathrm{SW}_{-i}(\theta_i) \\
&= 0
\end{aligned}$$

Intuitively, each agent i receives a payment equal to $\mathrm{SW}_{-i}(\theta_i)$ for its announced type, which is the expected social welfare effect on the other agents. To balance the budget each agent also pays an equal $1/(I \Leftrightarrow 1)$ share of the total payments made to the other agents, none of which depend on its own announced type.

Chapter 3

Computational Mechanism Design

The classic mechanism design literature largely ignores computational considerations. It is common to assume that agents can reveal their complete preferences over all possible outcomes (the *revelation principle*), and that the mechanism can solve an optimization problem to select the best outcome (e.g. the *Groves mechanisms*).

It is useful to take a "markets-as-computation" view of computational mechanism design. Our goal is to use a market-based method, such as an auction, to compute a social-welfare maximizing outcome to a distributed problem. This markets-as-computation view has received attention in computer science for a number of years, and in particular within artificial intelligence. Influential early work is the *Market Oriented Programming* (MOP) paradigm of Wellman [Wel93], which adopted economic equilibrium concepts as a technique to compute solutions to distributed optimization problems. Other classic early work includes that of Huberman & Clearwater [HC95] on a market-based system for air-conditioning control, and Sandholm [San93] on economic-based mechanisms for decentralized task-allocation amongst self-interested agents.

Early motivation for the market-based approach recognized that markets can provide quite efficient methods to solve distributed problems, prices for example can summarize relevant information about agents' local problems [Wel96]. The game-theoretic considerations of mechanism design were secondary to computational considerations. In recent years there has been increasing focus on game-theoretic issues, at first without much concern to computational tractability [RZ94], but later with attempts to integrate game-theoretic concerns and computational concerns [SL96, PU00b, Par01].

The tensions between classic game-theoretic solutions and tractable computational solutions soon become evident as one considers the application of mechanisms to difficult

distributed optimization problems, such as supply-chain procurement or bandwidth allocation.

Here is an outline of this chapter. Section 3.1 considers the different computational concerns in an implemented mechanism, looking at computation both at the agent and at the mechanism infrastructure level. I consider the consequences of the revelation principle for computational mechanism design. Section 3.2 focuses on the Generalized Vickrey Auction, and introduces methods to address both the cost of winner determination, the cost of complete information revelation, and to reduce communication costs. I also review work in the economics literature on mechanism design with limited communication structures.

3.1 Computational Goals vs. Game Theoretic Goals

Much of classic mechanism design is driven by the revelation principle (Section 2.3), which states informally that we only ever need to consider direct-revelation mechanisms. In a direct-revelation mechanism agents are restricted to sending a single message (the agent's preferences) to the mechanism, where that message makes a claim about the preferences of the agent over possible outcomes.

The revelation principle provides a very important theoretical tool, but is not useful in a constructive sense in difficult domains. The transformation assumed in the revelation principle from indirect mechanisms (e.g. an iterative auction) to direct-revelation mechanisms (e.g. a sealed-bid auction) assumes unlimited computational resources, both for agents in submitting valuation functions, and for the auctioneer in computing the outcome of a mechanism [Led89]. In particular, the revelation principle assumes:

— agents can compute and communicate their complete preferences

— the mechanism can compute the correct outcome with complete information about all relevant decentralized information in the system.

It can soon become impractical for an agent to compute and communicate its complete preferences to the mechanism, and for the mechanism to compute a solution to the centralized optimization problem. Direct-revelation mechanisms convert decentralized problems into centralized problems.

Yet, the revelation principle does have a vital role in the design of *all* mechanisms, both direct and indirect, sealed-bid and iterative. The revelation principle provides *focus*

to the design of iterative mechanisms. Taken along with the uniqueness of the Groves mechanisms amongst all efficient and dominant-strategy mechanisms (Section 2.4), then any efficient and iterative mechanism with incremental truth-revelation as a dominant strategy must compute the outcome of a Groves mechanism for the underlying preferences of agents. This is to avoid the existence of an equivalent direct-revelation mechanism for the mechanism that is outside of the class of Groves, which would be impossible.

It is useful to characterize the computation in a mechanism within two distinct levels:

1. At the *agent level*:

   (a) *Valuation complexity.* How much computation is required to provide preference information within a mechanism?

   (b) *Strategic complexity.* Must agents model other agents and solve game-theoretic problems to compute an optimal strategy?

2. At the *infrastructure level*:

   (a) *Winner-determination complexity.* How much computation is expected of the mechanism infrastructure, for example to compute an outcome given information provided by agents?

   (b) *Communication complexity.* How much communication is required, between agents and the mechanism, to compute an outcome?

Dominant strategy mechanisms, such as the Groves mechanisms, are efficient and strategy-proof mechanisms, giving them excellent strategic complexity. An agent can compute a dominant-strategy without modeling the other agents and without game-theoretic reasoning. However, the direct-revelation property of Groves mechanisms provides very bad (in fact *worst-case*) agent valuation complexity. An optimal bidding strategy requires that an agent determines its complete preferences over all possible outcomes. Complete information is required in all instances, even though it is often possible to solve a particular instance with incomplete information, with an interactive solution such as that provided in equilibrium solutions. The winner-determination complexity of Groves mechanisms in combinatorial domains also limits their applicability as problems get large; e.g. winner-determination in the combinatorial allocation problem is NP-hard.

Approaches to resolve this tension between game-theoretic and computational properties include:

- *Approximation methods.* Compute approximate outcomes based on agent strategies, and make connections between the accuracy of approximation and game-theoretic properties of the mechanism.

- *Distributed computation.* Move away from a centralized model of mechanism implementation towards models of decentralized computation to compute the outcome of a mechanism, based on information about agent preferences.

- *Special cases.* Identify tractable special cases of more general problems, and restrict the implementation space to those tractable special cases.

- *Compact preference representation languages.* Provide agents with expressive and compact methods to express their preferences, that avoid unnecessary details, make structure explicit, perhaps introduce approximations, and make the problem of computing optimal outcomes more tractable.

- *Dynamic mechanisms.* Instead of requiring single-shot direct-revelation, allow agents to provide incremental information about their preferences for different outcomes and solve easy problem instances without complete information revelation.

The challenge is to make mechanisms computationally feasible without sacrificing useful game-theoretic properties, such as efficiency and strategy-proofness.

3.2 Computation and the Generalized Vickrey Auction

The Generalized Vickrey Auction (GVA) is a classic mechanism with many important applications in distributed computational systems [NR01, WWWMM01]. As described in Section 2.4, the GVA is a strategy-proof and efficient mechanism for the combinatorial allocation problem, in which there are a set of items, $\mathcal{G}$, and a set of agents, $\mathcal{I}$, and the goal is to compute a feasible allocation of items to maximize the total value across all agents. Agents report values $\hat{v}_i(S)$ for each bundle $S \subseteq \mathcal{G}$, and the GVA computes an optimal allocation based on reported values and also solves one additional problem with each agent taken out of the system to compute payments.

From a computational perspective the GVA presents a number of challenges:

- *Winner determination is NP-hard.* Winner determination in the GVA is NP-hard, equivalent to the maximum weighted set packing problem. The auctioneer must solve the winner-determination problem once with all agents, and then once more with each agent removed from the system to compute payments.

- *Agents must* ***compute*** *values for an exponential number of bundles of items.* The GVA requires complete information revelation from each agent. The valuation problem for a single bundle can be hard [Mil00a], and in combinatorial domains there are an exponential number of bundles to consider.

- *Agents must* ***communicate*** *values for an exponential number of bundles of items.* Once an agent has determined its preferences for all possible outcomes it must communicate that information to the auctioneer. In addition to the network resource cost, this might be undesirable from a privacy perspective.

A number of proposals exist to address each of these problems, surveyed below. The first problem, concerning the computational complexity of the auctioneer's winner determination problem, has received most attention. In comparison, the second problem, concerning the complexity on participants to determine their preferences has received considerably less attention. Exceptions include the brief discussion of bidding programs in Nisan [Nis00], and the recent progress that has been made on dynamic mechanisms [WW00, Par99, PU00b].

This dynamic approach includes my *i*Bundle mechanism, and recent extensions to compute Vickrey payments. *i*Bundle is an iterative combinatorial auction mechanism, able to compute efficient allocations without complete information revelation from agents. The information savings follow from an *equilibrium-based interactive solution concept*, in which the efficient allocation is computed in an equilibrium between the auctioneer and the agents. In addition to terminating without complete information revelation in realistic problem instances, agents in *i*Bundle can compute optimal strategies without solving their complete local valuation problems.

3.2.1 Winner-Determination: Approximations and Distributed Methods

Possible ideas to address the complexity of winner-determination in the GVA include introducing approximation, identifying and restricting to special-cases, and distributed methods that seek to involve decentralized agents in the mechanism's computational problem.

In each case, as new methods are used to compute the outcome of a Groves mechanism, it is important to consider the effect on the strategy-proofness of the mechanism. There are two reasons to be concerned about the loss of strategy-proofness with an approximate Groves mechanism:

— agents can now benefit from game-theoretic reasoning, which makes their strategic bidding problem more difficult

— the effect of agents misrepresenting their preferences might further decrease the allocative-efficiency.

Approximations

Simply replacing the optimal algorithm in the GVA with an approximation algorithm does not preserve strategy-proofness. Recall that the utility to agent i in the GVA for reported preferences $\hat{\theta}_i$, is:

$$u_i(\hat{\theta}_i) = v_i(k^*(\hat{\theta}_i, \hat{\theta}_{-i}), \theta_i) + \sum_{j \neq i} v_j(k^*(\hat{\theta}_i, \hat{\theta}_{-i}), \hat{\theta}_j) \Leftrightarrow h_i(\hat{\theta}_{-i})$$

where $h_i(\cdot)$ is an arbitrary function over the reported preferences $\hat{\theta}_{-i} = (\hat{\theta}_1, \ldots, \hat{\theta}_{i-1}, \hat{\theta}_{i+1}, \ldots, \hat{\theta}_I)$ of the other agents.

Agent i chooses to announce preferences $\hat{\theta}_i$ to make $k^*(\hat{\theta}_i, \hat{\theta}_{-i})$ solve:

$$\max_{k \in \mathcal{K}} v_i(k, \theta_i) + \sum_{j \neq i} v_j(k, \hat{\theta}_j) \qquad (*)$$

Truth-revelation is a dominant strategy in the GVA because $k^*(\hat{\theta}_i, \hat{\theta}_{-i})$ solves this problem with $\hat{\theta}_i = \theta_i$.

With an approximate winner-determination algorithm the auctioneer selects outcome $\hat{k}(\hat{\theta}_i, \hat{\theta}_{-i})$, which might not equal $k^*(\hat{\theta}_i, \hat{\theta}_{-i})$. A rational agent will now announce a type $\hat{\theta}_i$ to try to make the approximation algorithm solve (*):

$$\hat{k}(\hat{\theta}_i, \theta_{-i}) = k^*(\theta_i, \theta_{-i})$$

In other words, the agent would like to make the approximation algorithm select the best possible outcome, given its *true preferences* and the announced preferences of the other agents, and this might perhaps be achieved by *manipulating* its inputs to the algorithm to "fix" the approximation. Truth-revelation is a dominant strategy within a Groves mechanism if and only if an agent cannot improve on the outcome computed by the mechanism's algorithm by misrepresenting its own preferences. This observation leads to useful characterizations of *necessary properties* for an approximation algorithm to retain strategy-proofness within a Groves mechanism.

Tennenholtz *et al.* [TKDM00] introduce a set of sufficient (but not necessary) axioms for an approximation algorithm to retain strategy-proofness. The most important axiom essentially introduces the following requirement (which the authors also refer to as "1-efficiency"):

$$v_i(\hat{k}(\theta_i, \hat{\theta}_{-i}), \theta_i) + \sum_{j \neq i} v_j(\hat{k}(\theta_i, \hat{\theta}_{-i}), \hat{\theta}_j) \geq v_i(\hat{k}(\hat{\theta}_i, \hat{\theta}_{-i}), \theta_i) + \sum_{j \neq i} v_j(\hat{k}(\hat{\theta}_i, \hat{\theta}_{-i}), \hat{\theta}_j),$$

for all $\hat{\theta}_i \neq \theta_i,\ \hat{\theta}_{-i},\ \theta_i$.

In words, an agent cannot improve the solution with respect to a particular set of inputs $(\theta_i, \hat{\theta}_{-i})$ by unilaterally misrepresenting its own input θ_i. Strategy-proofness follows quite naturally from this condition, given that a rational agent will only misrepresent its preferences to improve the quality of the solution (for reported preferences from other agents and the agent's true preferences) computed by the mechanism. An interesting open question is the degree to which these axioms restrict the efficiency of an approximation algorithm, for a particular class of algorithms (e.g. constant factor worst-case approximations, etc.).

Nisan & Ronen [NR00] take a different approach and define conditions on the *range* of an approximation algorithm, and require the algorithm to be *optimal* in its range—a condition they refer to as *maximal-in-range* —for strategy-proofness with approximate winner-determination algorithms. The conditions are necessary and sufficient.

The maximal-in-range condition states that if $\mathcal{K}' \subseteq \mathcal{K}$ is the range of outcomes selected by the algorithm (i.e. $k \in \mathcal{K}'$ implies there is some set of agent preferences for which the approximation algorithm $\hat{k}(\theta) = k$), then the approximation algorithm must compute the

best outcome in this restricted range for all inputs.

$$\hat{k}(\theta) = \max_{k \in \mathcal{K}'} \sum_{i \in \mathcal{I}} v_i(k, \theta_i)$$

for all $\theta \in \Theta$, and for some fixed $\mathcal{K}' \subseteq \mathcal{K}$.

Intuitively, strategy-proofness follows because the Groves mechanism with this rule implements a Groves mechanism in the reduced space of outcomes $\mathcal{K}'$. An agent cannot improve the outcome of the allocation rule by submitting a corrupted input because there is no reachable outcome of better quality. Nisan & Ronen partially characterize the necessary inefficiency due to the dual requirements of approximation and strategy-proofness, and claim that all truthful mechanisms with approximate algorithms have "unreasonable" behavior, for an appropriate definition of unreasonableness.

Lehmann *et al.* [LOS99] consider strategy-proof and approximate implementations for a special case of the combinatorial allocation problem, with *single-minded* bidders that care only about one bundle of items. Perhaps surprisingly, winner-determination remains NP-hard even in this very restricted problem by reduction from weighted set packing. Lehmann *et al.* allows the payment rules to change from those in a Groves scheme, and propose a set of sufficient axioms for strategy-proofness in their problem. The axioms apply to properties of the allocation rule *and* the payment rule. The most important condition for strategy-proofness is the "critical" condition, which states that an agent's payment must be independent of its bid and "minimal", closely following the intuition behind the incentive-compatibility of the Vickrey-Clarke-Groves scheme. Lehmann *et al.* propose a greedy allocation rule and a payment scheme that satisfies their axioms, which together comprise a strategy-proof mechanism. Extending to "double-minded" agents, the authors prove that there are *no* strategy-proof payment rules compatible with their greedy allocation method.

Nisan & Ronen [NR01] present an interesting algorithmic study of mechanism design for a task allocation problem, with a non efficiency-maximizing objective (and therefore outside of the application of Groves mechanisms). The objective in the task allocation problem is to allocate tasks to *minimize the makespan*, i.e. the time to complete the final task. Individually, each agent wants to minimize the time that it spends performing tasks. Nisan & Ronen present sufficient conditions for the strategy-proofness of a mechanism, and consider the class of approximation algorithms that satisfy those conditions. The

important axioms are *independence*, i.e. the payment to agent i does not depend on its own bid, and *maximization*, i.e. the mechanism must compute an outcome that maximizes the benefit when each agent reports its true ability to perform tasks. For a particular class of constant-factor approximation algorithms, the authors compute a lower-bound on the degree-of-approximation for which strategy-proofness is possible. Nisan & Ronen continue to show that a *randomized* mechanism can improve this best-case approximation factor.

Distributed Methods

Nisan & Ronen [NR00] propose an innovative "second chance" mechanism, which aims to combine distributed challenges by agents with an approximate winner-determination algorithm. The second-chance mechanism builds on the intuition that agents will manipulate an Groves mechanism built around an approximation algorithm if they can improve the solution by misrepresenting their own preferences, given the announcements of other agents.

The second-chance mechanism provides a method to allow agents to improve the outcome of the algorithm without also running the risk that agents will make the solution worse, for example if they make bad predictions about the algorithm or about the announcements from other agents. Agents submit a claim $\hat{\theta}_i$ about their preferences, as in the standard Groves mechanism, and also an *appeal function*, which can be viewed as an additional heuristic algorithm for the winner-determination problem. The appeal function provides an alternative set of inputs for the auctioneer's existing algorithm, i.e. $l : \Theta_1 \times \ldots \times \Theta_I \rightarrow \Theta_1 \times \ldots \times \Theta_I$, such that $l(\hat{\theta})$ provides a new set of preferences for each agent. Given reported types $\hat{\theta}$, the mechanism solves the optimization problem with its approximation algorithm once with the inputs $\hat{\theta}$, and once with each appeal set of inputs, e.g. $l_i(\hat{\theta})$ for the appeal function of agent i, *evaluating the outcomes in terms of the reported types* $\hat{\theta}$.

Intuitively, providing an appeal function allows each agent to try to fix the approximate nature of the auctioneer's winner-determination algorithm without needing to adjust its reported preferences away from its true preferences. Each agent will try to submit an appeal function to improve the system-wide (reported) value of the chosen solution. "Feasible truthfulness" of the second-chance mechanism is demonstrated, defined for a suitable restriction on the rationality of agents (see below).

The appeal functions are very complex and require a high degree of insight on the part of agents. Nisan & Ronen note that the agents themselves could be required to compute the results of their appeal function. The mechanism therefore can be viewed as a method to use decentralized computation to improve the performance of an approximate winner-determination algorithm. It is also suggested that agents be given the chance to learn the characteristics of the approximation algorithm, to enable them to generate good appeal functions. Another idea is to integrate successful appeals progressively into the heuristic, to improve its base performance.

Rothkopf *et al.* [RPH98] had earlier proposed decentralized computation approaches, with "challenges" issued to agents to improve the quality of the auctioneer's solution. Brewer [Bre99] also proposes a market mechanism to decentralize computation to agents.

Bounded-Rational Implementation

Returning to the concept of feasible truthfulness, there is one one sense in which bounded-rationality can *help* in mechanism design.

Nisan & Ronen [NR00] introduce the concept of a *feasible best-response* and a *feasible dominant action*. A feasible best-response is an agent's utility-maximizing action across a restricted set of all possible actions, known as the agent's *knowledge set*. The knowledge set is a mapping from the actions of other agents to a subset of an agent's own possible actions. An action is then *feasible dominant* if it is the best-response in an agent's knowledge set for all possible actions of other agents. This is a very similar concept to the maximal-in-range idea introduced as an axiom for strategy-proofness with approximate winner-determination algorithms.

Given this concept of feasible dominance, one might design mechanisms in which the strategies that perform better than truth-revelation are in small measure compared to all possible strategies, to make an agent require a lot of "knowledge" to have a non truth-revealing dominant strategy, or perform a lot of computation.

One can also interpret the *myopic best-response* strategy, adopted in my own work [Par99, PU00a], from the perspective of a bounded-rational agent. Certainly the assumption of myopia considerably simplifies an agent's problem, as it does not need to reason about the effect of its bids in the current round on future prices or on the strategies of other agents.

An interesting idea for future work is to design mechanisms that cannot be manipulated unless an agent can solve an NP-hard computational problem; i.e. use the bounded-ratonality of an agent to make it provably too difficult to manipulate a mechanism.

Special-Cases and Structure

Finally, let us consider the role of tractable special-cases of winner-determination. Rothkopf *et al.* [RPH98], Nisan [Nis00] and de Vries & Vohra [dVV00] characterize tractable special-cases, identifying restrictions on the types of bundles that can receive bids and/or the types of valuation functions agents can express over bundles (see Section 4.5). The approach is to restrict an agent's bidding language to induce only tractable winner-determination problems.

Ideally, a restricted bidding language can support tractable winner-determination without preventing agents reporting their true valuation functions. In this case the GVA mechanism can be applied without any loss in either strategy-proofness or efficiency. However, as soon as one imposes a restriction on agents' bids there is a risk that efficiency and strategy-proofness will be compromised. If an agent cannot represent its true valuation function with the restricted bidding language, then its rational strategy is to report an approximate value that leads to the best outcome for its true preferences, and force the mechanism to select the best solution from the set reachable from the restricted range of inputs. This ability to improve the outcome through non-truthful bidding leads to a loss in strategy-proofness, for example because the agent will now need to predict the strategies of other agents. The tradeoff between approximate bidding languages, incentive-compatibility, and efficiency appears to have received little attention.

Graphical tree representations, such as the Expected Utility Networks [MS99], allow an agent to capture independence structure within its preferences in much the same way as Bayes-Nets provide compact representations of conditional probabilities in suitable problems. In addition to providing quite compact and natural representations for participants, these structured approaches may allow tractable winner-determination and payment rules, that exploit the structure to solve problems without explicitly computing values for individual bundles.

3.2.2 Valuation Complexity: Bidding Programs and Dynamic Methods

The Groves mechanisms are direct-revelation mechanisms, requiring that every agent reports its complete preferences over all possible outcomes. In application to large combinatorial problems Groves mechanisms can fail because of the *bounded-rationality* of agents, and the *complexity of local valuation problems.* The valuation problem for a single bundle can be hard, and in combinatorial domains there are an exponential number of different bundles to consider.

Consider an application to a distributed package delivery problem, with agents competing for the delivery jobs. Agents represent delivery companies, and may need to solve hard local optimization problems to compute their costs to perform different bundles of jobs; each bundle may require that the agent computes an optimal schedule for its fleet of vehicles.

In these types of combinatorial problems a mechanism must not require an agent to report its complete valuation function. In addition, an agent must be able to compute its optimal strategy without computing its complete valuation function. It is not helpful to require less information if the agents must still compute values for all bundles to provide that information. In Chapter 8 I introduce a *bounded-rational compatible* characterization of auctions. The theory of bounded-rational compatibility precisely captures this idea that an agent can participate in an auction without performing unnecessary valuation work. In a bounded-rational compatible auction an agent can compute its equilibrium strategy with an approximate valuation function, at least in some problem instances.

Two interesting approaches to reduce information revelation in mechanisms for combinatorial allocation problems are:

(1) Retain the direct-revelation structure, but provide a *high-level bidding language* (or "bidding program") to allow an agent to "represent and define" its local problem without explicitly solving its local problem in all possible scenarios.

(2) Implement a *dynamic mechanism*, that requests information incrementally from agents and computes the optimal allocation and Vickrey payments without complete information revelation.

The first method may be helpful when specification is easier than valuation, i.e. it is easier for an agent to define how it determines its value for a bundle of items than it is to explicitly compute its value for all possible bundles. The second method may be

helpful when the iterative procedure terminates without complete information revelation by agents, and when an agent can provide incremental information without computing its complete valuation function. Let us consider each in turn.

Bidding Programs and High Level Bidding Languages

In choosing a bidding language for a mechanism there is a tradeoff between the ease with which an agent can represent its local preferences, and the ease with which the mechanism can compute the outcome. Nisan [Nis00] describes the *expressiveness* of a language, which is a measure of the size of a message for a particular family of valuation functions, and the *simplicity* of a language, which is a measure of the complexity involved in interpreting a language and computing values for different outcomes.

A natural starting point in combinatorial auctions is the XOR bidding language, (S_1, p_1) XOR (S_2, p_2), which essentially allows an agent to enumerate its value for all possible sets of items. This bidding language is simple to interpret, in fact given a bid b in the XOR language, the auctioneer can compute the value $b(S)$ for any bundle in polynomial time [Nis00]. However, this bidding language is not very expressive. An obvious example is provided with a *linear valuation* function, $v(S) = \sum_{x \in S} v(x)$. XOR bids for this valuation function are exponential in size (explicitly enumerating the value for all possible bundles) [Par99]. In comparison, an OR bidding language (S_1, p_1) OR (S_2, p_2), which states that the agent wants S_1 or S_2 or both, has a linear-space representation of this valuation function.

Nisan observes that other combinations, such as XOR-of-OR languages and OR-of-XOR languages, allow compact representations of certain preference structures and make tradeoffs across expressiveness and compactness. Nisan proposes an OR* bidding language, which is expressive enough to be able to represent arbitrary preferences over discrete items, and as compact a representation as both OR-of-XOR and XOR-of-OR representations. However, Nisan provides an example with no polynomial-size representation even with the OR* language.

The expressiveness of a bidding language, or the compactness of representations that it permits, becomes even more important when one considers the agent's underlying valuation problem.

Suppose that an agent must solve an NP-hard constrained optimization problem $[P]$ to compute its value for a set of items, with objective function g and constraints C. In the

XOR representation the agent must solve this problem $[P]$ once for every possible input $S \subseteq \mathcal{G}$, i.e. requiring an exponential number of solutions to an NP-hard problem. Now consider an alternative bidding language, that allows the agent to send the specification of its optimization problem, i.e. $[P] = (g, C)$ directly to the auctioneer. Strategy-proofness is not affected (assuming the agent can trust the mechanism to interpret this bidding language faithfully), but the agent saves a lot of value computation.

In general, we might consider a language in which the agent can send a "bidding program" to the auctioneer, that the auctioneer will then execute as necessary to compute an agent's value for different subsets of items [Nis00]. This is really just the extreme limit of the *revelation principle*: rather than requiring an agent to solve its local problem and compute its value for all possible outcomes, simply allow the agent to send the local problem specification directly to the auctioneer.

From the perspective of the bidding agent this approach simplifies its valuation problem whenever the specification of its local problem is simpler than actually computing its value for all possible outcomes. A bidding program allows an agent to feed that specification directly to the auctioneer.

From the perspective of the auctioneer, this is an even more centralized solution than providing a complete valuation function, and has worse-still privacy implications. The bidding program approach shifts the valuation computational burden from agents to the auctioneer. Notice for example that if the bidding program provides only "black box" functionality, e.g. $b : 2^{\mathcal{G}} \rightarrow \mathbb{R}$, the mechanism *must* compute $b(S)$ for all $S \subseteq \mathcal{G}$ (unless other consistency rules such as free disposal apply to an agents' values) to compute the efficient solution. However, if the bidding program, or language, provides a richer functionality— for example allowing efficient pruning "the value $b(S')$ on all bundles $S' \prec S$ is less than $b(S)$"; or computing approximate values "the value of $b(S)$ is between $[a, b]$"; or best-response "the bundle that maximizes $b(S) \Leftrightarrow p(S)$ at those prices is S_1" —then the total valuation work performed by the auctioneer can be less than that required by agents with the XOR bidding language. Savings of this kind can be realized within an algorithm that makes explicit use of these types of query structures.

Let me outline some serious limitations of the bidding program model in some domains:

- The specification problem can be as difficult as the valuation problem. In particular, the assumption above is that a *single* specification allows an agent to compute its

value for all bundles. In many problems the agent might need to collect additional information, consult human experts etc., to form a model with which to determine the value of each bundle. As a concrete example, consider the FCC spectrum auction. For any particular set of licenses a bidder might need to construct a new business model, to determine its value, and this can require costly and time-consuming information gathering, conference calls, and modeling efforts [Mil00a].

- Local valuation problems might not be well formed, the agent might not be able to provide a clear description of the method with which the value of each alternative is determined. This is a particular concern in systems in which human experts must be consulted to determine values.

- The size of the specification of a problem might be too large to transmit to the mechanism. Perhaps computing the value for a bundle requires access to a large database of information?

- Value and sensitivity of information. In a supply-chain example, will IBM really be happy to release the methods that it uses to take procurement decisions? This information has considerable value to a competitor.

- Trust. Can the agent trust the auctioneer to faithfully execute its bidding program? There might be a role for *verification mechanisms* to enable an agent to verify the value computation performed by a mechanism. Harkavy *et al.*[HTK98] and Naor *et al.* [NPS99] provide an introduction to some ideas from secure distributed computation that can be used in auction environments.

Dynamic Methods.

An alternative approach is to "open up" the algorithm for computing the outcome of the GVA, and involve agents dynamically in the computational process. It is easy to construct examples in which it is not necessary to have complete information about agents' valuation problems to compute and verify the outcome of the auction. A few simple examples are described at the end of this section. A well structured dynamic method might ask agents for *just enough* information to enable the mechanism to compute and verify the outcome.

A dynamic mechanism may elicit the following types of approximate information from agents:

— *ordinal information*, i.e. "which bundle has highest value out of S_1, S_2 and S_3?"

— *approximate information*, i.e. "is your value for bundle S_1 greater than 100?"

— *best-response information*, i.e. "which bundle do you want at prices $p(S)$?"

— *equivalence-set information*, i.e. "is there an item that is substitutable for A?"

In addition to solving realistic problem instances without complete information revelation, it is also important that dynamic methods allow an agent to respond to requests for information with an approximate solution to its own valuation function. Notice that in each of the preceding examples an agent can respond without first computing its exact value for all bundles.

Examples: Complete Information is Not Necessary

Examples 1–3 are simple problems instances in which the optimal allocation and the Vickrey payments can be computed without complete information from agents. Although there is no consideration of agent incentives at this stage, a well structured iterative auction can compute optimal outcomes without complete information from agents *and* provide incentives for agents to reveal truthful information.

Example 1. Single-item auction with 3 agents, and values $v_1 = 16, v_2 = 10, v_3 = 4$. The Vickrey outcome is to sell the item to agent 1 for agent 2's value, i.e. for 10. Instead of information $\{v_1, v_2, v_3\}$ it is sufficient to know $\{v_1 \geq 10, v_2 = 10, v_3 \leq 10\}$ to compute this outcome.

Example 2. Consider a combinatorial auction problem in which we ask every agent for the bundle that maximizes their value. If the response from each agent is non-overlapping, as illustrated in Figure 3.1 then we cam immediately compute the outcome of the GVA. The efficient allocation is to give each agent its favorite bundle; every agent gets its value-maximizing bundle so there can be no better solution. The Vickrey payments in this example are zero, intuitively because there is no competition between agents. We do not need any information about the value of an agent for any other bundles, and we do not need even need an agent's value for its favorite bundle.

Example 3. Consider the simple combinatorial allocation problem instance in Table 3.1, with items A, B and agents 1, 2, 3. The values of agent 1 for item B and bundle AB are stated as $a \leq b$ and $b \leq 15$, but otherwise left undefined. Consider the following cases:

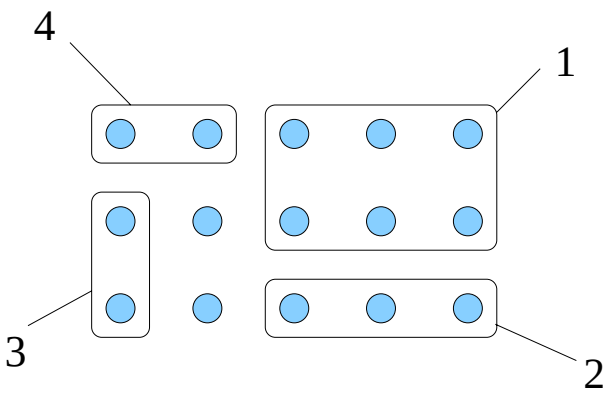


Figure 3.1: A simple combinatorial allocation problem. Each disc represents an item, and the selected bundles represent the bundles with maximum value for agents 1, 2, 3 and 4. In this example this is sufficient information from agents to compute the efficient solution (and the Vickrey payments).

[$a < 5$] In this case the GVA assigns bundle AB to agent 3, with $V^* = 15$, $(V_{-3})^* = \max[10+a, b]$, so that the payment for agent 3 is $p_{\text{vick}}(3) = 15 \Leftrightarrow (15 \Leftrightarrow \max[10+a, b]) = \max[10+a, b]$. It is sufficient to know $\{a \leq 5, b \leq 15, \max[10+a, b]\}$ to compute the outcome.

[$a \geq 5$] In this case the GVA assigns item B to agent 1 and item A to agent 2, with $V^* = 10 + a$, $(V_{-1})^* = 15$, and $(V_{-2})^* = 15$. The payment for agent 1 is $p_{\text{vick}}(1) = a \Leftrightarrow (10+a \Leftrightarrow 15) = 5$ and the payment for agent 2 is $p_{\text{vick}}(2) = 10 \Leftrightarrow (10+a \Leftrightarrow 15) = 15 \Leftrightarrow a$. It is sufficient to know $\{a, b \leq 15\}$ to compute the outcome.

Notice that it is not necessary to compute the *value* of the optimal allocation $\mathbf{S}^*$ to compute Vickrey payments; we only need to compute the allocation to each agent. Consider Example 1. We can compute the optimal allocation (give the item to agent 1) with information $v_1 \geq \{v_2, v_3\}$, and without knowing the exact value of v_1. Also, it is not even necessary to compute V^* and $(V_{-i})^*$ to compute Vickrey payments because common terms cancel. In Example 1, it is enough to know the value of v_2 to compute agent 1's Vickrey payment because the value of v_1 cancels: $p_{\text{vick}}(1) = v_1 \Leftrightarrow \Delta_{\text{vick}}(1) = v_1 \Leftrightarrow (v_1 \Leftrightarrow v_2) = v_2$.

Useful Properties of Iterative Auctions

Iterative price directed auctions, such as ascending-price auctions, present an important class of dynamic mechanisms. In each round of the auction the auctioneer announces prices

	A	B	AB
Agent 1	0	a	b
Agent 2	10	0	10
Agent 3	0	0	15

Table 3.1: Agent values in Example 3.

on the items, or bundles of items, and a *provisional allocation* (which agent is currently receiving which items). A reasonable bidding strategy for an agent is *myopic best-response*, which is simply to bid for the items that maximize its utility at the prices. Although myopic best-response is in general not the optimal *sequential strategy* for an agent, it can be made a Bayesian-Nash equilibrium of an iterative auction by computing Vickrey payments at the end of the auction (see Chapter 7).

Useful properties of iterative auctions include:

- Iterative auctions can solve realistic problems without complete information from agents. Consider an ascending-price auction for a single item. It is sufficient that the two agents with the highest value bid in each round, the other agents do not need to bid and can sit back and watch the price rise, without providing *any* information. Implicit information is provided by not responding to prices.

- Agents can follow myopic best-response without computing exact values for all bundles. For example, an agent can follow a best-response bidding strategy in a price-directed iterative auction with lower and upper bounds on its values for bundles. Myopic best-response only requires that an agent bids for the bundle(s) with maximum utility (value - price) in each round. This utility-maximizing set of bundles can be computed by refining the values on individual bundles until the utility of one or more bundles dominates all other bundles.

- The information requested dynamically in each round of an auction (implicitly, via the new prices and the bidding rules of the auction) is quite natural for agents (and people) to provide. The auction does not ask agents to make mysterious comparisons across different bundles, but rather lets agents consider their best-response (local utility-maximizing strategy) given the new prices.

*i*Bundle [Par99, PU00a], introduced in Chapter 5, is an ascending-price combinatorial auction. Agents can adjust their bids in response to bids placed by other agents, and the

auction eventually terminates in competitive equilibrium. *i*Bundle solves the problem in Figure 3.1 in one round with myopic best-response agent strategies, because every agent will bid for its value-maximizing bundle in response to zero prices and every agent will receive a bundle in the provisional allocation. In fact, *i*Bundle is provably efficient with myopic best-response agent strategies.

3.2.3 Communication Costs: Distributed Methods

Shoham & Tennenholtz [ST01] explore the communication complexity of computing simple functions within an auction-based algorithm (i.e., with self-interested agents with private information). Essentially, the authors propose a method to compute solutions to simple functions with minimal communication complexity. Communication from the auctioneer to the agents is free in their model, while communication from agents to the auctioneer is costly. Given this, Shoham & Tennenholtz essentially provide incentive schemes so that each agent i announces its value v_i by sending a single bit to the mechanism whenever the price in an auction is equal to this value. *Max* and *min* functions can be computed with a single bit from agents, and any function over n agents can be computed in n bits, which is the lower information-theoretic bound.

Feigenbaum *et al.* [FPS00] investigate cost-sharing algorithms for multicast transmission, in which a population of consumers sit on the nodes of a multicast tree. Each user has a value to receive a shared information stream, such as a film, and each arc in the multicast tree has an associated cost. The mechanism design problem is to implement the multicast solution that maximizes total user value minus total network cost, and shares the cost across end-users. Noting that budget-balance, efficiency, and strategy-proofness are impossibility in combination the authors compare the computational properties of a Vickrey-Clarke-Groves *marginal cost* (MC) mechanism (efficient and strategy-proof) and a Shapley value (SH) mechanism (budget-balanced and coalitional strategy-proof).

A distributed algorithm is developed for MC, in which intermediate nodes in the tree receive messages, perform some computation, and send messages to their neighbors. The method, a bottom-up followed by a top-down traversal of the tree, computes the solution to MC with *minimal* communication complexity, with exactly two messages sent per link. In comparison, there is no method for the SH mechanism with efficient communication complexity. All solutions are *maximal*, and require as many messages per link as in a naive

centralized approach. Hence, communication complexity considerations lead to a strong preference for the MC mechanism, which is not budget-balanced. The study leaves many interesting open questions; e.g. are all budget-balanced solutions maximal, and what are the game-theoretic properties of alternative strategy-proof minimal mechanisms?

The economic literature contains a few notable models of the effect of limited communication and agent bounded-rationality in mechanism design, and in systems of distributed decision making and information processing. This work is relevant here, given the focus in my dissertation on computational mechanism design and in particular on the costs of complete information revelation.

In the theory of teams [MR72], Radner and Marschak provide a computational account of the organization of management structures and teams, considering in particular the efficient use of information within a decentralized organization. One important assumption made in the theory of teams is that all agents share a common goal (e.g. profit), no attention is given to the incentives of agents. The goal is to compare the efficiency (decision quality) of different information structures under the assumption that each structure will be used optimally. The theory of teams proposes a two-step method to measure the effectiveness of a particular organizational structure: (1) find the optimal mode of functioning given a structure and compute the efficiency; (2) subtract the costs of operation. The second step in this methodology has not been done because there has traditionally been no good way to assess the *cost* of communication. One method suggested to side-step this problem is to compare the performance of different communication structures for a fixed number of messages. The work of Feigenbaum et al. [FPS00] certainly starts to integrate communication complexity analysis into mechanism design.

Radner [Rad87] compares the efficiency of four classic models of resource allocation, and asks which is the minimal sufficient structure to compute efficient solutions. Extensions to consider agent incentives are also discussed.

Recently, Radner [Rad92, Rad93] has considered a decision-theoretic model of a firm, in which managers are modeled as bounded-rational decision makers, able to perform some information processing and communicate. The model considers distributed decision problems in which agents must perform local computation with local information because of bounded-rationality and limited computation. One useful concept proposed by Radner is that of a "minimally efficient" network, which is the minimal communication network

(e.g. in terms of the number of links) that does not introduce delay the decentralized decision making of agents.

Green & Laffont [GL87] consider the impact of limited communication on the performance of incentive-compatible mechanisms. Starting with direct-revelation mechanisms, which assume that agents can transmit information messages that are sufficiently detailed to describe fully all their private information, Green & Laffont consider the effect of reducing the "dimensionality" of an agent's message space. In their abstract model the decision problem is to select an $x \in \mathbb{R}^n$, an agent's preferences are $\theta \in \mathbb{R}^m$, and the communication space is $R \in \mathbb{R}^l$. The authors characterize the effect of reducing the message dimension l, while trying to maintain incentive-compatibility and decision optimality.

There is a well developed theory on the minimal communication complexity required to implement efficient allocations [Hur72, MR74, Rei74]. Mount & Reiter compare the communication requirements at the equilibrium of different market structures, in which communication cost is measured in terms of the size of the message space that is used in a mechanism. However, most models compare the costs in equilibrium, without consider communication costs along the adjustment process, and without any attention to the computation cost on agents and on the mechanism infrastructure [Mar87]. A central question in the literature is: what is the minimal equilibrium message size required to implement a particular social choice function? Classic results argue that the "competitive mechanism", which implements allocations in equilibrium (the mechanism announces a set of prices and agents self-select which bundles they will consume), is informationally efficient. This provides a good theoretical basis for the attention to equilibrium solutions to distributed combinatorial optimization problems in this dissertation. Of course, I also carefully consider information revelation in the adjustment process as well as in equilibrium, in addition to the computational complexity of the agents and the auctioneer.

Chapter 4

Linear Programming and Auction Design

Mechanism design proposes the Vickrey-Clarke-Groves mechanism as a solution for the combinatorial allocation problem. In fact, this mechanism is essentially the *only* mechanism with the critical properties of strategy-proofness and allocative-efficiency. However, we have identified a number of inherent computational problems with Groves mechanisms:

- Every agent must provide its value for every bundle to the auctioneer.
- The auctioneer must solve multiple NP-hard problems to compute the outcome of the auction.

The Groves mechanisms are *centralized* solutions to a decentralized optimization problem. They address the incentive issues in systems with distributed agents with private information but fail to address important computational issues.

Naive approaches to address the auctioneer's computational problem will often break the strategy-proofness of the mechanism, in addition to reducing the allocative-efficiency of the solution. One method discussed in Chapter 3 suggests computing approximate solutions to the winner-determination problems, and computing Vickrey payments with the approximate solutions. Strategy-proofness is lost as soon as the mechanism does not compute the efficient allocation, the allocation that maximizes the reported values of agents. Instead an agent should try to misrepresent its valuation function *in just the right way* to make the auctioneer compute the optimal winner-determination solution *despite* its approximate algorithm. In other words agents should try to compensate for the approximation within the mechanism.

Similarly, while it is possible to restrict a bidding language such that the mechanism's winner-determination problem is tractable (see Table 4.4), as soon as the restricted expressiveness of the language forces the agent to submit an approximate report of its valuation

function, the agent must reason about *which approximation* will lead to the auctioneer computing a solution to the winner-determination problem that is maximal for the agent's true valuation function.

The motivation for *iterative* combinatorial auctions is to address the first computational problem, that of agent valuation work, *and* retain the useful game-theoretic properties of *strategy-proofness* and *efficiency*. In many problems it is quite unrealistic to assume that an agent can compute its value for all possible combinations of items, as is required in the single-shot VCG mechanism. Iterative combinatorial auctions provide *interactive* solutions, hopefully requesting *just enough* information from agents to compute the efficient allocation and the Vickrey payments. The uniqueness of Groves mechanisms (amongst direct-revelation mechanisms) implies via the revelation principle that any iterative solution to the combinatorial allocation problem with these desirable game-theoretic properties *must* compute the payments in the Vickrey-Clarke-Groves mechanism.

My approach is to first *assume* a simple bidding strategy for agents in each round of an iterative auction. The strategy, *myopic best-response*, need not be game-theoretically rational for an agent. However, this assumption allows a strong connection between linear programming theory, in particular primal-dual algorithms, and iterative auction design. Chapters 4 and 5 introduce primal-dual algorithm CombAuction, and its auction equivalent *i*Bundle, which computes efficient allocations with myopic best-response agent strategies. The prices computed in the dual solution have an economic interpretation, as the *competitive equilibrium* prices. Later, in Chapters 6 and 7, I present an extended primal-dual method, VickAuction, and an experimental auction design, *i*Bundle Extend&Adjust, to compute Vickrey payments and the efficient allocation with myopic best-response agent strategies. Vickrey payments make myopic best-response a *sequentially rational* strategy for an agent in equilibrium with myopic best-response from other agents— justifying my earlier assumption. This "LP + myopic best-Response + Vickrey" approach appears to provide a compelling methodology for the design of iterative mechanisms with useful game-theoretic properties.

Bertsekas [Ber87] had earlier proposed a primal-dual algorithm Auction for the *assignment problem*, which is a special case of the combinatorial allocation problem. Auction has a natural interpretation as an ascending-price auction, but does not compute Vickrey payments and does not have any useful incentive-compatibility properties. As discussed

in Section 4.7, authors such as Demange *et al.* [DGS86] and Ausubel [Aus97, Aus00], have proposed iterative auctions that compute both the efficient allocation and Vickrey payments for special-cases of the combinatorial allocation problem (CAP). In *i*Bundle I extend Bertsekas' primal-dual methodology to solve the CAP, without placing any restrictions on agent preferences, but for the moment without computing Vickrey payments (see Chapters 6 and 7 for this extension). *i*Bundle implements a primal-dual algorithm CombAuction, which computes solutions to linear program formulations of the CAP introduced in Bikchandani & Ostroy [BO99].

The extended auction, *i*Bundle Extend&Adjust, implements a primal-dual algorithm, VickAuction for a new linear program formulation of the Vickrey payments. VickAuction provably computes the efficient allocation and Vickrey payments with best-response information from agents. Computational results in Chapter 7 demonstrate that the experimental auction, *i*Bundle Extend&Adjust, which is an interpretation of primal-dual method VickAuction, computes Vickrey payments over a suite of problem instances. A full proof of the extended auction awaits a proof of termination of its final phase (see Chapter 7).

The outline of this chapter is as follows. Section 4.1 presents a brief description of *i*Bundle. Section 4.2 provides background on linear programming theory and primal-dual algorithms. Section 4.3 relates primal-dual methods with allocation problems, and considers price-adjustment methods and competitive equilibrium. The English auction provides a simple primal-dual example. Section 4.4 provides a hierarchy of linear programming formulations for the combinatorial allocation problem. Section 4.5 also outlines the tractable special-cases of the combinatorial allocation problem, and to provide practical interpretations as much as possible. This falls naturally within this chapter because the tractable special cases can all be understood within linear programming theory.

Section 4.6 describes CombAuction, a primal-dual algorithm for the combinatorial allocation problem. *i*Bundle, introduced in the next chapter, is a straightforward auction interpretation of CombAuction. Finally, Section 4.7 compares the characteristics and properties of CombAuction and *i*Bundle with earlier iterative auction mechanisms.

4.1 Overview: The *i*Bundle Auction

*i*Bundle [Par99, PU00a] is an ascending-price combinatorial auction, in which prices are maintained on bundles of items and agents can bid for bundles of items directly. In this section I give only a high-level description of the auction. Full details are presented in the next chapter. The description here is included to give some context to the primal-dual method, CombAuction, introduced to solve the CAP.

Allowing bids on bundles of items allows an agent to express a "I only want A if I also get B" type of constraint, that is observed to be important in many applications (see Chapter 1).

*i*Bundle maintains an ask price, $p_i(S) \geq 0$ for every bundle $S \subseteq \mathcal{G}$ and every agent $i \in \mathcal{I}$. This is the minimal price that an agent must bid for that bundle in the current round of the auction. The prices may be non-linear, $p_i(S) \neq \sum_{j \in S} p_i(j)$, and may be non-anonymous, $p_i(S) \neq p_j(S)$. In practice it is not necessary to explicitly price every bundle, prices are explicitly maintained on a subset of bundles (those which receive unsuccessful bids) and can be computed on any bundle as necessary. The auction also maintains a provisional allocation, $S = (S_1, \ldots, S_I)$ in each round. This is adjusted across rounds in response to agents' bids until the auction terminates, when it is implemented as the final allocation.

The key components of *i*Bundle are the bidding language, the winner-determination rules, the price-update rules, and the termination conditions. Agents place exclusive-or bids for bundles, e.g. S_1 XOR S_2, to indicate than an agent wants either all items in S_1 or all items in S_2 but not both S_1 and S_2. Each bid is associated with a bid price, which must be at least the ask price for the bundle. The auctioneer collects bids and computes a provisional allocation to maximize revenue given the bids. If every agent that placed a bid receives a bundle the auction terminates. Prices are initially anonymous, with $p_{\text{ask},i}(S) = p_{\text{ask},j}(S) = p(S)$, but a simple rule introduces price discrimination dynamically, as necessary to terminate with competitive equilibrium prices and an efficient allocation. The price on a bundle is increased to $\epsilon > 0$ above the highest bid price for the bundle from any unsuccessful agent in the current round (an agent not in the provisional allocation). *i*Bundle terminates when every agent that bids at the current prices receives a bundle in the provisional allocation.

The final prices are competitive equilibrium prices, and the final allocation efficient, if

agents follow a myopic best-response bidding strategy, bidding for bundles that maximize their utility given the prices in each round.

4.2 Linear Programming Theory

First, I provide a brief review of basic results in linear programming. See Papadimitriou & Steiglitz [PS82] for a text book introduction, and Chandru's excellent survey papers [CR99b, CR99a] for a modern review of the literature on linear programming and integer programming.

Consider the linear program:

$$\begin{aligned} & \max\ c^T x \\ \text{s.t.}\quad & Ax \leq b \\ & x \geq 0 \end{aligned} \qquad \text{[P]}$$

where A is a $m \times n$ integer matrix, $x \in R^n$ is a n-vector, and c and b are $n\Leftrightarrow$ and m-vectors of integers. vectors are column-vectors, and notation c^T indicates the *transpose* of vector c, similarly for matrices. The primal problem is to compute a feasible solution for x that maximizes the value of the objective function.

The dual program is constructed as:

$$\begin{aligned} & \min\ b^T y \\ \text{s.t.}\quad & A^T y \geq c \\ & y \geq 0 \end{aligned} \qquad \text{[D]}$$

where $y \in R^m$ is a m-vector. The dual problem is to compute a feasible solution for y that minimizes the value of the objective function.

Let $V_{\text{LP}}(x) = c^T x$, the value of feasible primal solution x, and $V_{\text{DLP}}(y) = b^T y$, the value of feasible dual solution y.

The *weak duality theorem* of linear programming states that the value of the dual always dominates the value of the primal:

THEOREM 4.1 (weak-duality). *Given a feasible primal solution x with value $V_{\mathrm{LP}}(x)$ and a feasible dual solution y with value $V_{\mathrm{DLP}}(y)$, then $V_{\mathrm{LP}}(x) \leq V_{\mathrm{DLP}}(y)$.*

PROOF. Solution x is feasible, so $Ax \leq b$. Solution y is feasible, so $A^T y \geq c$. Therefore, $x \leq A^T b$ and $y \geq Ac$, and $c^T x \leq c^T A^T b = b^T AC \leq b^T y$, and $P \leq D$. ▮

The *strong duality theorem* of linear programming states that primal and dual solutions are optimal if and only if the value of the primal equals the value of the dual:

THEOREM 4.2 (strong-duality). *Primal solution x^* and dual solution y^* are a pair of optimal solutions for the primal and dual respectively, if and only if x^* and y^* are feasible (satisfy respective constraints)* and $V_{\mathrm{LP}}(x^*) = V_{\mathrm{DLP}}(y^*)$.

The strong-duality theorem of linear programming can be restated in terms of *complementary-slackness* conditions (CS for short). Complementary-slackness conditions expresses logical relationships between the values of primal and dual solutions that are necessary and sufficient for optimality.

DEFINITION 4.1 [complementary-slackness] Complementary-slackness conditions constrain pairs of primal and dual solutions. *Primal* CS conditions state $x^T(A^T y \Leftrightarrow c) = 0$, or in logical form:

$$x_j > 0 \Rightarrow A^j y = c_j \tag{P-CS}$$

where A^j denotes the jth column of A (written as a row vector to avoid the use of transpose). *Dual* CS conditions state $y^T(Ax \Leftrightarrow b) = 0$, or in logical form:

$$y_j > 0 \Rightarrow A_i x = b_i \tag{D-CS}$$

where A_i denotes the ith row of A.

The strong-duality theorem can be restated as the *complementary-slackness theorem*:

THEOREM 4.3 (complementary-slackness). *A pair of feasible primal, x, and dual solutions, y, are primal and dual optimal if and only if they satisfy the complementary-slackness conditions.*

PROOF. P-CS iff $x^T(A^T y \Leftrightarrow c) = 0$, and D-CS iff $y^T(Ax \Leftrightarrow b) = 0$. Equating, and observing that $x^T A^T y = y^T Ax$, we have P-CS and D-CS iff $x^T c = y^T b$, or $c^T x = b^T y$. The LHS is the value of the primal, $V_{\mathrm{LP}}(x)$, and the RHS is the value of the dual, $V_{\mathrm{DLP}}(y)$. By the strong duality theorem, $V_{\mathrm{LP}}(x) = V_{\mathrm{DLP}}(y)$ is a necessary and sufficient condition for the solutions to be optimal. ▌

4.2.1 Primal-Dual Algorithms

Primal-dual is an algorithm-design paradigm that is often used to solve combinatorial optimization problems. A problem is first formulated both as a primal and a dual linear program. A primal-dual algorithm searches for feasible primal and dual solutions that satisfy complementary-slackness conditions, instead of searching for an optimal primal (or dual) solution directly. Primal-dual can present a useful algorithm-design paradigm for combinatorial optimization problems. Instead of solving a single hard primal solution, or a single hard dual solution, a primal-dual approach solves a sequence of restricted primal problems. Each restricted primal problem is often much simpler to solve than the full primal (or dual) problem [PS82].

Primal-dual theory also provides a useful conceptual framework for the design of iterative combinatorial auctions. Prices represent a feasible dual solution, and bids from agents allow a search for a primal solution that satisfies complementary-slackness conditions. If the current solution is suboptimal there is enough information available to adjust dual prices in the right direction. Complementary-slackness conditions provide the key to understanding how it is possible to compute and verify optimal solutions without complete information: it is sufficient to just verify that a feasible solution satisfies CS conditions. Primal-dual algorithms are consistent with the decentralized information inherent in distributed agent-based systems. Optimality reduces to a test of feasibility and complementary-slackness, which is available from agent bids, rather than the direct solution of a primal problem, which requires information about agent valuation functions.

A standard primal-dual formulation maintains a feasible dual solution, y, and computes a solution to a *restricted primal problem*, given the dual solution. The restricted primal is formulated to compute a primal solution that is both feasible and satisfies CS conditions with the dual solution. In general this is not possible (until the dual solution is optimal),

and a relaxed solution is computed. The restricted primal problem is typically formulated to compute this relaxed solution in one of two ways:

1. Compute a feasible primal solution x' that minimizes the "violation" of complementary-slackness conditions with dual solution y.

2. Compute a primal solution x' that satisfies complementary slackness conditions with dual solution y, and minimizes the "violation" of feasibility constraints.

Method (1) is more useful in the context of iterative auction design because it maintains a feasible primal solution, which becomes the *provisional allocation* in the auction, i.e. a tentative allocation that will be implemented only when the auction terminates. The restricted primal problem can be solved as a *winner-determination problem*. I show that computing the allocation that maximizes revenue given agent bids (the solution to winner-determination) is a suitable method to minimize the violation of CS conditions between the prices and the provisional allocation in each round iBundle. Prices in each round of an auction define the feasible dual solution, and agent best-response bids provide enough information to test for complementary-slackness and adjust solutions towards optimality.

As discussed in the introduction to this chapter, I first assume myopic best-response, but later justify this assumption with an extension to compute Vickrey payments at the end of the auction in addition to the efficient allocation (see Chapters 6 and 7).

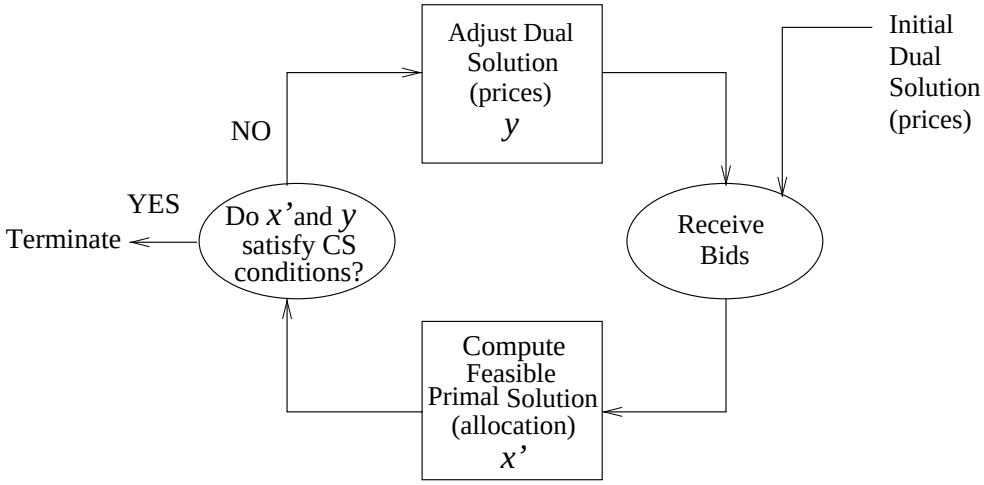


Figure 4.1: A primal-dual interpretation of an auction algorithm.

A primal-dual based auction method has the following form (see Figure 4.1):

1. Maintain a feasible dual solution ("prices").

2. Compute a feasible primal solution ("provisional allocation") to minimize violations with complementary-slackness conditions given agents' bids.

3. Terminate if all CS conditions are satisfied ("are the allocation and prices in competitive equilibrium?")

4. Adjust the dual solution towards an optimal solution, based on CS conditions and the current primal solution ("increase prices based on agent bids")

4.3 Allocation Problems

Let us consider the particular form of an *allocation problem*, in which there are a set of discrete items to allocate to agents, and the goal is to maximize value. We assume quasi-linear preferences, and use *utility* to refer to the difference between an agent's value for a bundle and the price. The primal and dual allocation problems can be stated as follows:

DEFINITION 4.2 [allocation problem: primal] The primal allocation problem is to allocate items to agents to maximize the sum value over all agents, such that no item is allocated to more than one agent.

DEFINITION 4.3 [allocation problem: dual] The dual allocation problem is to assign *prices* to items, or bundles of items, to minimize the sum of (i) each agents' maximum utility given the prices, over all possible allocations; *and* (ii) the maximum revenue over all possible allocations given the prices.

Clearly, without information on agents' values the auctioneer cannot compute an optimal primal or an optimal dual (because of term *(i)* in the dual). However, under a reasonable assumption about agents' bidding strategies (myopic best-response) the auctioneer can verify complementary-slackness conditions between primal and dual solutions, and adjust prices and the allocation towards optimal solutions.

An auction interpretation of the complementary-slackness conditions can be stated as follows:

DEFINITION 4.4 [allocation problem: CS conditions] The CS between a feasible primal solution to an allocation problem, x, and a feasible dual solution, prices p, are:

(CS-1) Agent i receives bundles S_i in the provisional allocation if and only if the bundle maximizes its utility given the prices, and has non-negative utility.

(CS-2) The provisional allocation $S = (S_1, \ldots, S_I)$ is the revenue-maximizing allocation given the prices.

Left deliberately vague at this stage is the exact *structure* of the prices. In a combinatorial allocation problem these might need to be non-linear and non-anonymous prices to support the optimal allocation. Similarly, the revenue-maximization concept must be defined with respect to a particular linear program formulation. Note also that CS-2 is not automatically satisfied with a provisional allocation computed to maximize revenue given agents' bids. CS-2 makes a stronger claim, that the provisional allocation must maximize revenue over *all possible* allocations given the current ask prices, not just over all allocations consistent with bids.

Primal-dual auction analysis requires the following assumption about agent strategies:

DEFINITION 4.5 [myopic best-response] A myopic best-response bidding strategy is to bid for all items or bundles of items that maximize utility at the current prices.

Best-response bids provide enough information to test CS-1, because the best-response of an agent is precisely those bundles that maximize an agent's utility given the current prices. For any feasible primal solution, the auctioneer can test CS-2 because that only requires price information.

The restricted primal has a natural auction interpretation:

DEFINITION 4.6 [auction restricted-primal problem] Given best response bids from each agent allocate bundles to maximize revenue, breaking ties in favor of including more agents in the provisional allocation.

Note well that a bundle is only allocated to an agent in the restricted primal problem if the agent bids for that bundle. This restriction ensures that CS-1 is satisfied for that agent, given the definition of myopic best-response. CS-2 is satisfied with careful price-adjustment rules, such that prices are increased "slowly enough" that the revenue-maximizing allocation can always be computed from agent bids.

Given myopic best-response, the termination condition, which tests for complementary-slackness between the provisional allocation and the prices, must check that CS-1 holds

for every agent. This is achieved when every agent to submit a bid receives a bundle in the provisional allocation, i.e. in competitive equilibrium.

Our interest is in solving the CAP, which is most immediately formulated as an integer program (see Section 4.4). In order to apply primal-dual methods it is essential that we have a linear program formulation of the CAP. We must be careful enough to use a strong enough formulation, such that the optimal solution is integral (0-1) and not fractional. The ideal situation is illustrated in Figure 4.2. The auction implements a primal-dual algorithm for a linear program that is strong enough to compute the optimal integer solution.

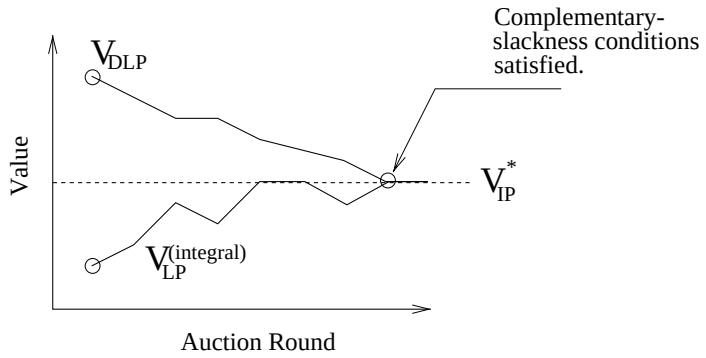


Figure 4.2: An auction-based primal-dual algorithm in which the linear program formulation is strong enough to eliminate all fractional solutions.

In comparison, consider Figures 4.3 (a) and (b), which illustrate a primal-dual algorithm and iterative auction method for a linear program that is not strong enough, and admits optimal fractional solutions. The primal-dual algorithm algorithm terminates with a fractional primal solution and value greater than the value of the best possible integer solution. The auction always maintains an integral primal solution (solving winner-determination to compute the provisional allocation), but can terminate with a primal solution that does not satisfy complementary-slackness conditions. Although the primal solution is perhaps optimal, its optimality cannot be assessed without CS information.

4.3.1 Price Adjustment

Left undefined at the moment, and the challenging part of primal-dual auction design, are the precise rules used to define price updates. The goal is to use information from agents' bids, and the current provisional allocation, to adjust prices towards an optimal dual solution— that will support an optimal primal solution. Primal-dual methods traditionally

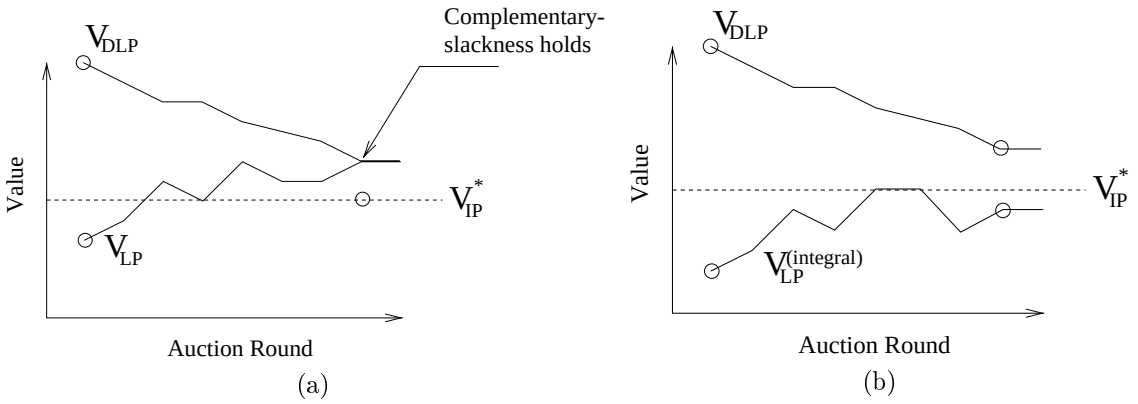


Figure 4.3: Primal-dual algorithm (a) and Primal-dual auction method (b) in which the linear program relaxation is too weak, and $V^*_{\text{LPR}} > V^*_{\text{IP}}$.

use the dual of the restricted primal to adjust the dual solution across iterations. A simpler method in allocation problems is to *increase prices on over-demanded items, or bundles of items.* The method can be explained both in terms of its effect on complementary-slackness conditions and in terms of its effect on the value of the dual solution.

The idea is to increase prices to: (a) maintain CS-2 in the next round and (b) move towards satisfying CS-1 for all agents.

PROPOSITION 4.1 (progress). *Progress is made towards satisfying CS-1 and CS-2 with the provisional allocation and the ask prices if: (1) the auctioneer increases prices on one or more bundles that receive bids in each round; and (2) the auctioneer increases prices by a small enough increment that best-response bids from agents continue to maximize revenue in the next round.*

CS-1 holds whenever every agent that bids receives a bundle in the provisional allocation. This is trivially achieved for high enough prices because no agent will bid, but we need to achieve this condition in combination with CS-2. The trick is to increase prices just enough to maintain revenue-maximization from bids CS-2 across all rounds. This is achieved in *i*Bundle by ensuring that myopic agents continue to bid for bundles at the new prices, i.e. increasing price on over-demanded bundles.

An alternative interpretation is that increasing prices on over-demanded items will *reduce* the *value of the dual*, making progress towards the optimal solution, see Figure 4.4. Recall that the value of the dual is the sum of the auctioneer's maximal revenue and each

agent's maximal utility at the current prices. A price increase will decrease the value of the dual if the increase in maximal revenue from the price increase is *less* than the decrease in total maximal utility summed across agents.

The auctioneer can achieve this effect of increasing revenue by less than the decrease in agent utility by selecting over-demanded items, or bundles of items, on which to increase the price. Suppose that two agents bid for bundle S_1, and that both agents have at least $\epsilon > 0$ more utility for that bundle than any other bundle at the current prices. Increasing the price on over-demanded bundle S_1 by ϵ will decrease the maximal utility of *both* agents by ϵ, for a decrease in dual value of 2ϵ. However, increasing the price on this one bundle by ϵ can increase the auctioneer's maximal revenue by at most ϵ. The result is that the net change in utility must a decrease of at least ϵ.

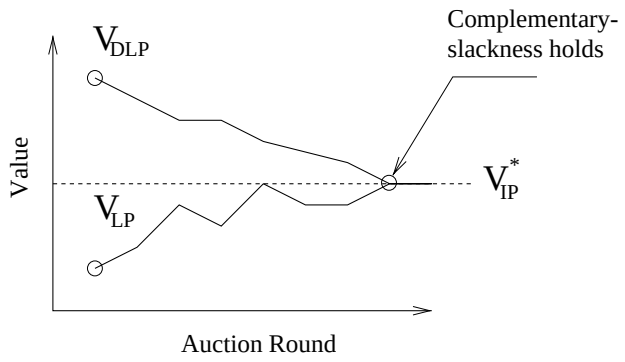


Figure 4.4: Primal-dual interpretation of an ascending-price auction.

4.3.2 Competitive Equilibrium

The optimal primal and dual solutions in an allocation problem correspond to a classic statement of *competitive equilibrium*.

Definition 4.7 [competitive equilibrium] Allocation S and prices p are in competitive equilibrium when:

(a) every agent receives a bundle in its best-response (utility maximizing) set

(b) the allocation maximizes the revenue for the auctioneer at the prices

The allocation in competitive equilibrium is efficient, by equivalence between competitive equilibrium and primal-dual optimality:

Theorem 4.4 (competitive equilibrium efficiency). *An allocation S is efficient if and*

only if there exists competitive equilibrium prices p, for an appropriate type of prices (e.g. linear, bundle, non-anonymous).

In the context of the combinatorial allocation problem Bikchandani & Ostroy [BO99] have characterized the structure on prices required for the existence of competitive equilibrium (and equivalently for integral solutions to linear program formulations of CAP). These formulations are introduced in Section 4.4 and discussed at length.

In some problems it is necessary that prices are both non-linear (bundle prices) and non-anonymous (different prices for the same bundle to different agents) to support a competitive equilibrium solution.

Wurman & Wellman [WW99, WW00] propose an alternative definition of competitive equilibrium, which is essentially complementary slackness condition CS-1 without CS-2. This relaxed condition is sufficient for the existence of equilibrium prices even without non-anonymous prices, but too weak to be able to claim that equilibrium prices imply an efficient allocation.

4.3.3 Example: The English Auction

The standard English auction illustrates the primal-dual framework for auction design. The English auction is an ascending-price auction for single items, where the price increases as long as more than one agent bids at the current price.

Let v_i denote agent i's value for the item. The single-unit resource allocation problem is:

$$\begin{aligned} & \max \sum_i v_i x_i && [\text{IP}_{\text{single}}] \\ \text{s.t.} \quad & \sum_i x_i \leq 1 \\ & x_i \in \{0,1\} \end{aligned}$$

where $x_i = 1$ if and only if agent i is allocated the item, i.e. the goal is to allocate the item to the agent with the highest value. This can be solved as a linear program, $[\text{LP}_{\text{single}}]$, relaxing the integral constraint

$$\begin{aligned} & \max \sum_i v_i x_i && [\text{LP}_{\text{single}}] \\ \text{s.t.} \quad & \sum_i x_i \leq 1 \\ & x_i \geq 0 \end{aligned}$$

and $V^*_{\text{LP}} = V^*_{\text{IP}}$, i.e. there is always an integral optimal solution to the relaxed problem. The dual formulation, $[\text{DLP}_{\text{single}}]$, is

$$\begin{aligned} & \min \pi && [\text{DLP}_{\text{single}}] \\ \text{s.t.} \quad & \pi \geq v_i, \quad \forall i \\ & \pi \geq 0 \end{aligned}$$

The complementary-slackness conditions are

$$\begin{aligned} \textstyle\sum x_i \geq 0 \quad &\Rightarrow \quad \pi = v_i, \qquad \forall i \\ \pi > 0 \quad &\Rightarrow \quad \textstyle\sum x_i = 1 \end{aligned}$$

The complementary-slackness conditions can be interpreted in terms of competitive equilibrium conditions on the allocation and the prices. An allocation and prices in a single-item auction are in competitive equilibrium, and the allocation is efficient, when:

(i) the item is sold to an agent, that agent bids for the item at the price, and no other agent bids for the item at the price.

or (ii) the item is sold to no agent, the price is zero, and no agent bids for the item.

It is straightforward to understand efficiency in these cases: in (i) the agent with the highest value receives the item; in (ii) no agent has a positive value for the item.

The English auction maintains price p on the item, initially $p = 0$. Agent i bids whenever $p < v_i$, and the provisional allocation sets $x_j = 1$ for one of the agents that bids in each round, and increases the price p whenever more than one agent bids.

Let the provisional allocation define a feasible primal solution, and the price define dual solution $\pi = \sum_i \max\{0, v_i \Leftrightarrow p\} + p$. This is feasible, $\pi \geq \max\{0, v_i \Leftrightarrow p\} + p \geq v_i$ for all agents i.

Assume that agents follow a myopic best-response bidding strategy, bidding for the item at the ask price whenever the price is below their value. The optimality of the English auction can be understood in two different ways:

- The English auction terminates with primal and dual solutions that satisfy CS-1 and CS-2.

  Clearly, CS-2 is satisfied throughout the auction because the item is always allocated to one of the agents. CS-1 is satisfied when the auction terminates. Let j indicate the only agent that bids at price p. Therefore $v_i \Leftrightarrow p \leq 0$ for all agents $i \neq j$ and $v_j \Leftrightarrow p \geq 0$ for agent j (because agents follow best-response bidding strategies), and $\pi = \sum_i \max\{0, v_i \Leftrightarrow p\} + p = \max\{0, v_j \Leftrightarrow p\} + p = v_j$.

- The value of the dual strictly decreases in each round of the auction. Let $m > 1$ equal the number of agents that bid in each round of the auction except the final round. For price increment ϵ, the sum maximal utility to the agents decreases by $m\epsilon$ and the maximal revenue to the auctioneer increases by ϵ, for a net change in π of $\Leftrightarrow(m \Leftrightarrow 1)\epsilon$.

In fact, the final price in the English auction approaches the Vickrey payment (i.e. the second-highest value) as the bid increment $\epsilon \to 0$. It follows that myopic-best response is a rational *sequential* strategy for an agent, in equilibrium with myopic best-response strategies from other agents (see Chapter 7 for a full discussion of the incentive properties of iterative Vickrey auctions).

4.4 Linear Program Formulations for the Combinatorial Allocation Problem

Primal-dual based auction methods require linear programming formulations of allocation problems. Bikchandani & Ostroy [BO99] have formulated a hierarchy of linear programs for the problem, introducing additional constraints to remove fractional solutions. Although it is always possible to add enough constraints to a linear program relaxation to make the optimal solution integral [Wol81a, Wol81b, TW81], the particular formulations proposed by Bikchandani & Ostroy are interesting because the constraints have natural interpretations as prices in the dual.

The hierarchy of linear program formulations, [LP_1], [LP_2], and [LP_3], all retain the set of integer allocations but prune additional fractional solutions. Each formulation introduces new constraints into the primal, with the dual problems [DLP_1], [DLP_2], and [DLP_3] containing richer price structures. For example, in [DLP_1] the prices on a bundle are linear in the price of items, i.e. $p(S) = \sum_{j \in S} p(j)$, where $p(j)$ is the price of item j in bundle S. Moving to [DLP_2], the price on a bundle can be non-linear in the price on items, and in [DLP_3] the price on a bundle can be different to different agents. Bikchandani & Ostroy prove that LP_3 solves all CAP instances, and demonstrate the existence of competitive equilibrium prices, even though they must sometimes be both non-linear and non-anonymous.

Solving the CAP with the high-level linear program formulations is likely to be less efficient computationally than direct search-based methods applied to the integer program formulation. Formulations [LP_2] and [LP_3] introduce an exponential number of additional primal constraints, and dual variables, effectively enumerating *all possible* solutions to the CAP. In comparison, search methods, such as branch-and-bound with LP-based heuristics, solve the problem with implicit enumeration and pruning.

However the formulations are very useful in the context of mechanism design and decentralized CAP problems. In Section 4.6 I present CombAuction, a primal-dual algorithm for the CAP, which

(a) computes optimal primal and dual solutions *without* complete information about agent valuation functions.

(b) computes optimal primal and dual solutions *without* complete enumeration of all primal constraints and/or dual variables.

In fact most of the computation within CombAuction occurs in winner determination, which solves the restricted primal problem in each round, and winner-determination itself is solved with a branch-and-bound search algorithm.

4.4.1 Integer Program Formulation

Introducing $x_i(S)$ to indicate that agent i receives bundle S the straightforward integer program, [IP], formulation of the combinatorial allocation problem is:

$$\max_{x_i(S)} \sum_S \sum_i x_i(S) v_i(S) \qquad \text{[IP]}$$

$$\text{s.t.} \quad \sum_S x_i(S) \leq 1, \quad \forall i \tag{IP-1}$$

$$\sum_{S \ni j} \sum_i x_i(S) \leq 1, \quad \forall j \tag{IP-2}$$

$$x_i(S) \in \{0,1\}, \quad \forall i, S$$

where $S \ni j$ indicates a bundle S that contains item j. The objective is to compute the allocation that maximizes value over all agents, without allocating more than one bundle to any agent (IP-1) and without allocating a single item multiple times (IP-2). Let V^*_{IP} denote the value of the optimal allocation.

4.4.2 First-order LP Formulation

LP_1 is a direct linear relaxation, which replaces the integral constraints $x_i(S) \in \{0,1\}$ with non-negativity constraints, $x_i(S) \geq 0$.

$$\max_{x_i(S)} \sum_S \sum_i x_i(S) v_i(S) \tag{LP_1}$$

$$\text{s.t.} \quad \sum_S x_i(S) \leq 1, \qquad \forall i \tag{LP_1-1}$$

$$\sum_{S \ni j} \sum_i x_i(S) \leq 1, \qquad \forall j \tag{LP_1-2}$$

$$x_i(S) \geq 0, \qquad \forall i, S$$

$$\min_{p(i),p(j)} \sum_i p(i) + \sum_j p(j) \tag{DLP_1}$$

$$\text{s.t.} \quad p(i) + \sum_{j \in S} p(j) \geq v_i(S), \qquad \forall i, S \tag{DLP_1-1}$$

$$p(i), p(j) \geq 0, \qquad \forall i, j$$

Prices $p(j)$ on items $j \in \mathcal{G}$ define a feasible dual solution, with the substitution $p(i) = \max_S \left\{ v_i(S) \Leftrightarrow \sum_{j \in S} p(j) \right\}$.

PROPOSITION 4.2 (first-order dual). *The value of the first-order dual is the sum of the maximal utility to each agent plus the total price over all items (this is the auctioneer's maximal revenue).*

	A	B	AB
Agent 1	0	0	3
Agent 2	2^*	0	2
Agent 3	0	2^*	2

Table 4.1: Problem 1.

	A	B	C	AB	BC	AC	ABC
Agent 1	60	50	50	200^*	100	110	250
Agent 2	50	60	50	110	200	100	255
Agent 3	50	50	75^*	100	125	200	250

Table 4.2: Problem 2.

The dual variables define linear prices, the price for bundle $S \subseteq \mathcal{G}$ is $p(S) = \sum_{j \in S} p(j)$. From Definition 4.7 the optimal dual solution defines competitive equilibrium prices if and only if a partition of items exists at the prices that allocates each agent a bundle in its utility-maximizing set and allocates every item with positive price exactly once.

Problem 1 in Table 4.1 can be solved with [LP$_1$]; $V^*_{\text{LP}_1} = V_{\text{IP}} = 4$. The optimal allocation is $x_2(A) = 1$ and $x_3(B) = 1$, indicated by *. To see that $V_{\text{LP}_1} \leq 4$, notice that dual prices $p(A) = p(B) = 1.6$ gives a dual solution with value $V_{\text{DLP}_1} = 0{+}0.4{+}0.4{+}3.2 = 4$. Remember that $V^*_{\text{LP}_1} \leq V_{\text{DLP}_1}$ for all dual solutions by the weak-duality theorem of linear programming. These are one set of competitive equilibrium prices.

However, in general the value $V^*_{\text{LP}_1} > V^*_{\text{IP}}$ and the optimal primal solution makes fractional assignments to agents. As an example of when [LP$_1$] fails, consider Problem 2 in Table 4.2. In this problem $V^*_{\text{LP}_1} = 300 > V^*_{\text{IP}} = 275$. The primal allocates fractional solution $x_1(AB) = 0.5, x_2(BC) = 0.5$ and $x_3(AC) = 0.5$, which satisfies constraints (LP$_1$-1) because $\sum S \ni j \sum_i x_i(S) \leq 1$ for all items $j \in \mathcal{G}$. Prices $p(A) = p(B) = p(C) = 100$ solve the dual problem DLP$_1$.

Kelso & Crawford [KC82] prove that gross-substitutes (GS) preferences are a sufficient condition for the existence of linear competitive equilibrium prices, such that $V^*_{\text{LP}_1} = V^*_{\text{IP}}$.

To define gross-substitutes preferences, let $D_i(p)$ define the demand set of agent i at prices p, i.e. the set of bundles that maximize its utility (value - price).

Definition 4.8 [gross-substitutes (GS)] For all price vectors p, p' such that $p' \geq p$, and all $S \in D_i(p)$, there exists $T \in D_i(p')$ such that $\{j \in S : p_j = p'_j\} \subset T$.

In words, an agent has GS preferences if an agent continues to demand items with the same price as the price on other items increases. If preferences are also *monotonic*, such that $v_i(S') \geq v_i(S)$ for all $S' \supseteq S$, then GS implies *submodular* preferences.

DEFINITION 4.9 [submodular preferences] Valuation function $v_i(S)$ is submodular if for all $S, T \subseteq \mathcal{G}$,

$$v_i(S) + v_i(T) \geq v_i(S \cup T) + v_i(S \cap T)$$

Submodularity is equivalent to a generalized statement of *decreasing returns*:

DEFINITION 4.10 [decreasing returns] Valuation function $v_i(S)$ has decreasing marginal returns if for all $S \subset T \subseteq \mathcal{G}$ and all $j \in \mathcal{G}$,

$$v_i(T) \Leftrightarrow v_i(T \setminus \{j\}) \leq v_i(S) \Leftrightarrow v(S \setminus \{j\})$$

In other words, the value of an item increases as it is introduced to larger sets of items. Subadditivity implies that the value for any bundle is no greater than the minimal sum of values for a partition of the bundle.

In fact, gross-substitutes preferences define the largest set of preferences that contain unit-demand preferences (see Definition 4.14) for which the existence of linear competitive equilibrium prices can be shown [GS99].

The rest of this section introduces two alternative linear program formulations of CAP, [LP_2] and [LP_3], due to Bikchandani & Ostroy [BO99].

4.4.3 Second-order LP Formulation

Introducing new constraints to the first-order linear program relaxation [LP_1] of [IP] gives a second-order linear program [LP_2] with dual [DLP_2]. The corresponding dual variables to the new primal constraints are interpreted as *bundle prices* within an auction-based primal-dual algorithm.

$$\max_{x_i(S),y(k)} \sum_S \sum_i x_i(S) v_i(S) \qquad [\text{LP}_2]$$

$$\text{s.t.} \quad \sum_S x_i(S) \leq 1, \qquad \forall i \qquad (\text{LP}_2\text{-}1)$$

$$\sum_i x_i(S) \leq \sum_{k \ni S} y(k), \qquad \forall S \qquad (\text{LP}_2\text{-}2)$$

$$\sum_k y(k) \leq 1 \qquad (\text{LP}_2\text{-}3)$$

$$x_i(S), y(k) \geq 0, \qquad \forall i, S, k$$

$$\min_{p(i),p(S),\pi} \sum_i p(i) + \pi \qquad [\text{DLP}_2]$$

$$\text{s.t.} \quad p(i) + p(S) \geq v_i(S), \qquad \forall i, S \qquad (\text{DLP}_2\text{-}1)$$

$$\pi \Leftrightarrow \sum_{S \in k} p(S) \geq 0, \qquad \forall k \qquad (\text{DLP}_2\text{-}2)$$

$$p(i), p(S), \pi \geq 0, \qquad \forall i, S$$

where $k \in K$ is a *partition* of items in set K, and $k \ni S$ indicates that bundle S is represented in partition k. A partition is a feasible "bundling" of items, e.g. $[A, B, C]$ or $[AB, C]$, etc., and K is the set of all possible partitions, e.g. $K = \{[A, B, C], [AB, C], [A, BC], \ldots, [ABC]\}$ in Problem 2 (Table 4.2).

Constraints (LP_2-2) and (LP_2-3) replace constraints (LP_1-1), and ensure that no more than one unit of every item is allocated. The dual [DLP_2] has variables $p(i)$, $p(S)$ and π, which correspond to constraints (LP_2-1), (LP_2-2) and (LP_2-3), and constraints (DLP_2-1) and (DLP_2-2) correspond to primal variables $x_i(S)$ and $y(k)$.

Dual variables $p(S)$ can be interpreted as bundle prices, and with substitution $p(i) = \max_S \{v_i(S) \Leftrightarrow p(S)\}$, i.e. the maximal utility to agent i at prices $p(S)$, and
$\pi = \max_{k \in K} \sum_{S \in k} p(S)$, i.e. the maximal revenue to the auctioneer at prices $p(S)$.

PROPOSITION 4.3 (second-order dual). *The value of the dual is the sum of the maximal utility to each agent with bundle prices* $p(S)$ plus *the auctioneer's maximal revenue over all feasible (and non-fractional) allocations at the prices.*

	A	B	AB
Agent 1	0	0	3^*
Agent 2	2	2	2

Table 4.3: Problem 3.

The dual variables correspond to bundle prices, $p(S)$, and the optimal dual solution defines competitive equilibrium prices (by Definition 4.7) if there is an allocation that gives each agent a bundle in its utility-maximizing set at the prices, and maximizes revenue to the auctioneer over all possible allocations.

With the additional constraints [LP_2] solves Problem 2. Allocation $x_1(AB) = x_2(BC) = x_3(AC) = 0.5$ is *not* feasible in [LP_2] because it is not possible to allocate $y(k_1) = y(k_2) = y(k_3) = 0.5$ for $k_1 = [AB, C], k_2 = [AC, B]$ and $k_3 = [AB, C]$ without violating constraint (LP_2-3) and without this we violate constraints (LP_2-2). [LP_2] solves Problem 2, with $V^*_{\text{LP}_2} = V^*_{\text{IP}} = 275$. An optimal dual solution is given by bundle prices $p = (50, 60, 75, 190, 200, 200, 255)$, with total agent maximal utility $10 + 0 + 0$ and maximal auctioneer revenue $75 + 190 = 265$, i.e. $V^*_{\text{DLP}_2} = 275$.

However, Problem 3 is an example that [LP_2] does not solve. The value of the optimal primal solution is $V^*_{\text{LP}_2} = 3.5$, which is greater than the value of the optimal feasible allocation, $V^*_{\text{IP}} = 3$. The primal allocates fractional bundles $x_1(AB) = 0.5$ and $x_2(A) = x_2(B) = 0.5$, which satisfies constraints (LP_2-2) and (LP_2-3) with $y(k_1) = y(k_2) = 0.5$ for partitions $k_1 = [AB, \emptyset]$ and $k_2 = [A, B]$. Prices $p(A) = 1.5, p(B) = 1.5, p(AB) = 3$ solves the dual problem DLP_2.

4.4.4 Third-order LP Formulation

Introducing new constraints to the second-order linear program relaxation [LP_2] of [IP] gives a third-order linear program [LP_3] with dual [DLP_3]. The corresponding dual variables to the new primal constraints are interpreted as *non-anonymous*, or *discriminatory* bundle prices, with different prices for the same bundle to different agents.

$$\max_{x_i(S),y(k)} \sum_S \sum_i x_i(S) v_i(S) \qquad [\text{LP}_3]$$

$$\text{s.t.} \quad \sum_S x_i(S) \leq 1, \quad \forall i \qquad (\text{LP}_3\text{-}1)$$

$$x_i(S) \leq \sum_{k \ni [i,S]} y(k), \quad \forall i, S \qquad (\text{LP}_3\text{-}2)$$

$$\sum_k y(k) \leq 1 \qquad (\text{LP}_3\text{-}3)$$

$$x_i(S), y(k) \geq 0, \quad \forall i, S, k$$

$$\min_{p(i),p_i(S),\pi} \sum_i p(i) + \pi \qquad [\text{DLP}_3]$$

$$\text{s.t.} \quad p(i) + p_i(S) \geq v_i(S), \quad \forall i, S \qquad (\text{DLP}_3\text{-}1)$$

$$\pi \Leftrightarrow \sum_{[i,S] \in k} p_i(S) \geq 0, \quad \forall k \qquad (\text{DLP}_3\text{-}2)$$

$$p(i), p_i(S), \pi \geq 0, \quad \forall i, S$$

where $k \ni [i, S]$ indicates that *agent-partition* $k \in K'$ contains bundle S designated for agent i. Variable $y(k)$ in [LP$_3$] corresponds to an *agent-partition* k, where the set of agent-partitions in Problem 3 is $K' = \{[(1, A), (2, B)], [(1, B), (2, A)], [(1, AB), (2, \emptyset)], [(1, \emptyset), (2, AB)]$. It is important to note that each agent can receive at most one bundle in a particular agent-partition.

The dual variables $p_i(S)$ that correspond to primal constraints (LP$_3$-2) are interpreted as *non-anonymous* bundle prices, price $p_i(S)$ is the price to agent i for bundle S. As before, substitutions $p(i) = \max_S \{v_i(S) \Leftrightarrow p_i(S)\}$, i.e. the maximal utility to agent i at individual prices $p_i(S)$, and $\pi = \max_{k \in K'} \sum_{[i,S] \in k} p_i(S)$, i.e. the maximal revenue to the auctioneer at prices $p_i(S)$ given that it can allocate at most one bundle at prices $p_i(S)$ to each agent i.

PROPOSITION 4.4 (third-order dual). *The value of the dual to [LP$_3$] is the sum of the maximal utility to each agent with bundle prices* $p_i(S)$ plus *the auctioneer's maximal revenue over all feasible allocations at the prices. In this case an allocation is feasible if it allocates no more than one bundle to each agent.*

The dual variables correspond to non-anonymous bundle prices, $p_i(S)$, and the optimal dual solution defines competitive equilibrium prices if there is an allocation of items that simultaneously gives each agent a bundle in its utility-maximizing set and maximizes the auctioneer's revenue, over all possible allocations that sell at most one bundle to each agent.

Bikchandani & Ostroy [BO99] prove this important theorem:

THEOREM 4.5 (integrality). *The optimal solution to linear program [LP_3] is always integral, and therefore an optimal solution to CAP, with* $V^*_{\mathrm{LP}_3} = V^*_{\mathrm{DLP}_3} = V^*_{\mathrm{IP}}$.

Therefore, there are always competitive equilibrium bundles prices for CAP, although these prices must be non-anonymous in some problems.

Consider Problem 3. Allocation $x_1(AB) = 0.5$ and $x_2(A) = x_3(B) = 0.5$ is *not* feasible in [LP_3] because $y(k_1) = y(k_2) = y(k_3) = 0.5$ for $k_1 = [(1, AB), (2, \emptyset)], k_2 = [(1, A), (2, B)]$ and $k_3 = [(1, B), (2, A)]$ violates constraint (LP_3-3), but without this constraints (LP_3-2) are violated. In this problem $V^*_{\mathrm{LP}_3} = V^*_{\mathrm{IP}} = 3$. To see this, consider bundle prices $p_1 = (0, 0, 2.5)$ and $p_2 = (2, 2, 2)$, for which the value of the dual is $0.5 + 0 + 2.5 = 3$. This proves that $V_{\mathrm{LP}_3} \leq 3$ by the weak-duality theorem of linear programming.

I will return to this hierarchy of linear-program formulations of the CAP in Section 4.6, when I introduce the CombAuction primal-dual algorithm. CombAuction constructs feasible primal and dual solutions to an appropriate linear program formulation, and adjusts the solution until complementary-slackness conditions are also satisfied.

4.5 Tractable Combinatorial Allocation Problems

The CAP is equivalent to the maximum weighted set packing problem (SPP), a well-studied problem in the operations research literature. In SPP there are a set of items, and a set of subsets each with non-negative weights, and the goal is to pack the items into sets to maximize total value, without using any item more than once. CAP can be reduced to SPP by introducing an additional "dummy item" for the XOR bids from each agent. de Vries & Vohra [dVV00] also note two closely related problems, the set partitioning problem (SPA), in which the goal is to select a set of subsets with minimal

cost that include all items at most once, and the set covering problem (SCP), in which the goal is to select a set of subsets with minimal cost that include all items at least once. Set covering problems find applications in railway crew-scheduling and airline scheduling, where items are flights/trains, and bundles represent possibility sets for individual workers. A considerable amount is known about the complexity of this class of problems.

A classic technique in combinatorial optimization theory is to relax an integer program to a linear one. Many tractable special cases follow by considering the conditions on the natural relaxation of the integer program that provide integer solutions. For example, one sufficient condition is that the linear program is *integral*, such that all extremal feasible points are integral, i.e. 0-1. In this case the integrality requirement can be dropped and the problem solved as a linear program in polynomial time. Restrictions on the constraint matrix, corresponding to restrictions on the kinds of subsets permitted in CAP, can provide this integrality property [dVV00].

Additional restrictions, for example on the size of bids, or on the valuation structure of bids, can also lead to tractable special cases. Given the connection with linear programming relaxations this is a good place to review known tractable special-cases in the literature. The results here are drawn from Rothkopf *et al.* [RPH98], de Vries & Vohra [dVV00], Nisan [Nis00], and earlier work due to Kelso & Crawford [KC82].

It is important to understand the characteristics of tractable special-cases of CAP because this knowledge can be leveraged within mechanism design, achieving tractable and strategy-proof solutions (see Section 3.2.1 in Chapter 3).

Restrictions on Structure of Bundles

Table 4.4 presents tractable instances of CAP that follow from restrictions on the types of bundles on which agents can submit bids. de Vries & Vohra note that the linear-ordering (or consecutive ones) condition implies that the constraint matrix satisfies *total unimodularity*,[1] and that the nested-hierarchical structure implies that the constraint matrix is *balanced*.[2] Nisan [Nis00] provides a proof-by-induction that the linear program has integral solutions in these cases, and also describes a method to combine two bid structures with the integral property into a single structure that retains the property.

[1] A matrix satisfies total unimodularity if the determinant of every square submatrix is 0, 1, or -1.

[2] A 0-1 matrix is balanced if it has no square submatrix of odd order with exactly two 1's in each row and column.

linear-order	ordering $G = (g_1, g_2, \ldots, g_n)$ every bid is for a contiguous sequence	[RPH98]
circular ones	also allow bids of form $g_n g_1 g_2$, etc.	[RPH98]
nested-hierarchical	for every two subsets of items S_1, S_2 that appear as part of any bid they are either disjoint or one contains the other	[RPH98]
or-singletons	bids for single-items	-
single-item bids	one item	-
bids for pairs of items	cardinality constraint on size of bids	[RPH98]
multi-unit, decreasing returns	identical items, each agent has decreasing value for each additional item	[Nis00]

Table 4.4: Tractable structure on bids

non-decreasing and supermodular two-types of agents	"increasing returns"	[dVV00]
gross-substitutes	"decreasing-returns"	[KC82]
unit-demand	agents only want one item	[Kuh55]
linear-additive	agents have linear values across items	[CK81]

Table 4.5: Constraints on valuation functions

Restrictions on Values on Bundles

Table 4.5 presents tractable instances of CAP that follow from restrictions on the value structure of agents bids. de Vries & Vohra [dVV00] note that the non-decreasing and supermodular preferences condition again provides the linear program relaxation of the CAP with integral solutions. Gross-substitutes were defined earlier in Definition 4.8 and have an intuitive interpretation as decreasing-returns, and also imply submodular preferences.

DEFINITION 4.11 [supermodular preferences] Bid function $b_i(S)$ is supermodular if for all $S, T \subseteq \mathcal{G}$,

$$b_i(S) + b_i(T) \leq v_i(S \cup T) + v_i(S \cap T)$$

The equivalence of supermodularity and increasing returns is well-known in the literature [GS99].

DEFINITION 4.12 [increasing returns] Bid function $b_i(S)$ has increasing marginal returns if for all $S \subset T \subseteq \mathcal{G}$ and all $j \in \mathcal{G}$,

$$b_i(T) \Leftrightarrow v_i(T \setminus \{j\}) \geq b_i(S) \Leftrightarrow v(S \setminus \{j\})$$

Note carefully that we can have any number of different types of submodular valuation functions, one from each agent, but only at most *two* different types of supermodular functions if the CAP problem is to be tractable. It is easier to solve a maximization problem, such as the CAP, with submodular (convex) objective functions than supermodular (concave) objective functions.

Exact Solutions

Rothkopf *et al.* [RPH98] also suggest a dynamic programming algorithm for CAP, which has run-time complexity independent of the number of bids actually placed, but quickly becomes intractable for large numbers of items, with scaling property $O(3^m)$ in the number of items m. Branch-and-bound search methods, either with AI-based heuristics [San99, FLBS99], or with linear-program based heuristics [ATY00] have also been studied for general CAP instances.

Approximate Solutions

The CAP is difficult to approximate, at least within a worst-case multiplicative factor. There is no polynomial time algorithm with a reasonable worst-case guarantee [Has99].

Approximation algorithms in the literature without this guarantee include a local-search approach [HB00], a simple "relax and round" method [Nis00], and iterative methods [FLBS99]. CombAuction can itself be viewed as an approximate algorithm for CAP. CombAuction provides a worst-case bound on the difference between the value of its solution and the value of the optimal solution. This error-term increases linearly with the minimal bid increment, which defines the rate at which prices are increased across rounds, while the number of rounds in the auction is inversely-proportional to the minimal bid increment. A larger bid increment reduces the number of rounds in the auction, reducing the number of winner-determination problems the auction must solve, in return for a loss in worst-case efficiency. Experimental results in Section 5.5.1 show the effectiveness of this approach.

4.6 CombAuction: A Primal-Dual Method for CAP

CombAuction is a primal-dual algorithm for the linear program models of CAP introduced in Section 4.4. The algorithm terminates with optimal primal and dual solutions to an appropriate level in the linear-program hierarchy, selecting the price structure dynamically to support the optimal allocation in equilibrium. In the next chapter I describe the *i*Bundle auction, which is an auction-based implementation of CombAuction for agents that follow myopic best-response bidding strategies.

In CombAuction the prices are linear and anonymous in special cases, non-linear and anonymous (bundle prices) in many problems, and non-linear and non-anonymous when that is necessary to strengthen the dual formulation of the CAP. The decision about anonymous vs. non-anonymous pricing is made dynamically during the algorithm, while non-linear prices are introduced whenever an agent bids for a bundle of items instead of individual items.

4.6.1 Description

Let $p_{\text{ask}}(S) \geq 0$ denote anonymous ask prices on bundles $S \subseteq \mathcal{G}$. Set the initial ask prices to zero, and use only anonymous prices at the start of the auction. The prices represent a feasible *dual solution* in each round of the algorithm. Non-anonymous prices are denoted $p_{\text{ask},i}(S)$, for the price on bundle S to agent i, and initially $p_{\text{ask},i}(S) = p_{\text{ask}}(S)$ for all agents and all bundles.

Let $\hat{S} = (\hat{S}_1, \ldots, \hat{S}_I)$ denote the provisional allocation, computed from agents' bids in each round to maximize revenue, where agent i is provisionally allocated bundle $\hat{S}_i$. Let $p_i^+(S)$ denote *personalized* ask prices to agent i, computed as follows:

$$p_i^+(S) = \begin{cases} p_{\text{ask},i}(S) & \text{, if } p_{\text{ask},i}(S) \leq v_i(S) \text{ and } S \neq \hat{S}_i \\ p_{\text{ask},i}(S) \Leftrightarrow \epsilon & \text{, otherwise} \end{cases}$$

The personalized ask price is the price that agent i can actually bid for bundle S. In words, an agent must bid the ask price unless the ask price is greater than its value or the agent receives the bundle in the current allocation, in which case it can bid ϵ below the ask price.

Figure 4.5 describes CombAuction. The basic variation, also known as CombAuction(d), introduces non-anonymous prices whenever agent bids are not "safe". Two other

COMBAUCTION
input: `agent values` $v_i(\cdot)$
`stop = false;` $\hat{S} = \emptyset$; $\mathbf{p}_{\text{ask}}(\cdot) = 0$; anon $= \mathcal{I}$;
`while (¬ stop) {`
  `update personalized prices` $p_i^+(S)$ `for every agent` i;
  `compute best-response set` $\text{BR}_i(\mathbf{p}_i^+)$ `for every agent` i;
  `compute partition of items` $\hat{S} = (\hat{S}_1, \ldots, \hat{S}_I)$ `to maximize revenue,`
    `subject to` $\hat{S}_i \in \text{BR}_i(\mathbf{p}_i^+)$`.;`
  `if (` $(\text{BR}_i(\mathbf{p}_i^+) = \emptyset)$ `or unchanged(`$\text{BR}_i(\mathbf{p}_i^+)$`) or (`$\hat{S}_i \neq \emptyset$`) ) for every` i
    `stop = true;`
  `else {`
    `for every` j `with ((`$\text{BR}_j(\mathbf{p}_j^+) \neq \emptyset$`) and (`$\hat{S}_j = \emptyset$`)) {`
      `if (` $(safe(\text{BR}_j(\mathbf{p}_j^+)))$ `and (` $j \in anon$`))`
        $\mathbf{p}_{\text{ask}}$ `= anon_update(`$\text{BR}_j(\mathbf{p}_j^+), \mathbf{p}_{\text{ask}}$`);`
      `else {`
        $\mathbf{p}_{\text{ask},j}$ `= nonanon_update(`$\text{BR}_j(\mathbf{p}_j^+), \mathbf{p}_{\text{ask},j}$`);`
        $anon = anon \setminus \{j\}$;
      }
    }
  }
}
output: `final allocation` $\hat{S}$`, final prices` $\mathbf{p}_{\text{ask},i}(\hat{S}_i)$.

Figure 4.5: The COMBAUCTION algorithm. Special cases: COMBAUCTION(2) sets $safe(\text{BR}_j(\mathbf{p}_j^+))$ true in every iteration; COMBAUCTION(3) sets $safe(\text{BR}_j(\mathbf{p}_j^+))$ false in every iteration.

important variations include COMBAUCTION(2), in which the safe condition is always assumed `true`, and COMBAUCTION(3), in which the safe condition is always assumed `false`. Labels (D), (2), and (3) correspond to "dynamic" non-anonymous pricing, second-order pricing (i.e. non-linear), and third-order pricing (i.e. non-linear and non-anonymous).

Prices are initially zero on all bundles, and identical across all agents such that $p_{\text{ask},i}(S) = p_{\text{ask}}(S)$ for all i and all S. At the start of each iteration each the new personalized prices $\mathbf{p}_i^+$ are computed for each agent. Next, each agent $i \in \mathcal{I}$ reports its best-response set $\text{BR}_i(\mathbf{p}_i^+)$. The best-response set is the set of all bundles and corresponding personalized prices that maximize utility to within $\epsilon > 0$:

$$\text{BR}_i(\mathbf{p}_i^+) = \{(S, p_i^+(S)) \mid v_i(S) \Leftrightarrow p_i^+(S) + \epsilon \geq \max(u_i^*(\mathbf{p}_i^+), 0)\ \}$$

where $u_i^*(\mathbf{p}_i^+) = \max_S v_i(S) \Leftrightarrow p_i^+(S)$, i.e. the maximal utility over all bundles at the current (personalized) prices. Constant $\epsilon > 0$ is the minimal bid increment, and controls

the rate at which prices are increased across rounds.

The provisional allocation $\hat{\mathbf{S}} = (\hat{S}_1, \ldots, \hat{S}_I)$, is computed from agents' bids to maximize revenue:

$$\max_{(S_1,\ldots,S_I)} \sum_{i \in \mathcal{I}} p_i^+(S_i) \qquad \text{(WD problem)}$$
$$\text{s.t.} \qquad (S_i, p_i^+(S_i)) \in \mathrm{BR}_i(\mathbf{p}_i^+)$$
$$S_i \cap S_j = \emptyset, \qquad \forall i, j$$

Prices are increased based on bids from agents not in the current provisional allocation. In each round the agents $i \in \mathit{anon}$ receive anonymous prices, $p_{\mathrm{ask}}(S)$, and the agents $i \notin \mathit{anon}$ receive non-anonymous prices, $p_{\mathrm{ask},i}(S)$, which can charge a different price for the same bundle to different agents. Initially every agent receives anonymous prices, and $\mathit{anon} = \mathcal{I}$.

The price-update step depends on whether an agent's bids are *safe*. The *safe* condition is defined on a set of bundles $\mathcal{S}$ as follows:

$$\mathit{safe}(\mathcal{S}) = \neg \mathit{disjoint}(\mathcal{S}) \text{ or } (|\mathcal{S}| = 1)$$

where $\mathit{disjoint}(\mathcal{S})$ is true if there is at least one pair of bundles $S, T \in \mathcal{S}$ that are non-overlapping, such that $S \cap T = \emptyset$, and $|\mathcal{S}| = 1$ denotes a best-response set with only one bundle.

Price increases based on unsuccessful but *safe* bids from an agent j in *anon* are computed with update rule $\texttt{anon\_update}(\mathrm{BR}_j(\mathbf{p}_j^+), \mathbf{p}_{\mathrm{ask}})$, which increases the price $p_{\mathrm{ask}}(S)$ to $p_j^+(S) + \epsilon$ on all bundles S in agent j's best-response bid set, where constant $\epsilon > 0$ is the *minimal bid increment*. This price increase affects all agents in *anon*. Notice that if the agent's personalized bid price is ϵ below the current ask price, for example if an agent is repeating a bid for a bundle in the provisional allocation, then the price on this unsuccessful bid does *not* increase the price on the bundle.

In the first round that an agent's bids are unsuccessful and fail the *safe* condition the agent is removed from the anonymous set and faces individual prices in all future rounds. Price increases to an agent not in *anon* are based only on its own bids, and its bids never directly affect the prices to agents that remain in *anon*. Individual ask prices are initially set to the current anonymous prices. Price update rule $\texttt{nonanon\_update}(\mathrm{BR}_j(\mathbf{p}_j^+), \mathbf{p}_{\mathrm{ask},j})$

increases the price $p_{\text{ask},j}(S)$ to agent j to $p_j^+(S)+\epsilon$, for every bundle S in its best-response set. Again, this will only *increase* the price if the personalized price is not discounted by ϵ from the ask price.

CombAuction terminates when every agent with a non-empty best-response set *either* receives a bundle in the provisional allocation:

$$\text{BR}_i(\mathbf{p}_i^+) \neq \emptyset \quad \Rightarrow \quad \hat{S}_i \neq \emptyset \qquad \text{condition [T1]}$$

or submits the same best-response bids in two successive rounds (condition [T2]).

CombAuction(2) is the special-case of CombAuction for which the safety condition is assumed true in all rounds and all agents remain in the anonymous set. CombAuction(d) is the regular version of CombAuction, described above, in which non-anonymous prices are introduced dynamically. CombAuction(3) is a variation in which the safety condition is assumed false in all rounds and all agents face individual prices in every round, i.e. the set $anon = \emptyset$ from round 1.

Discussion

The personalized prices ensure that price increases do not "overshoot" the values of agents that receive a bundle in the efficient allocation. Suppose for example that the allocation problem has a single item, and there are two agents with the same value for the item. It is essential that when the agents eventually drive the price above their value, one of the agents, i.e. the agent with the item in the final provisional allocation, is able to repeat a bid for the item at the previous price— just before it was forced to high.

Allowing agents to submit a best-response bid set, that can include more than one bundle, and is accurate to within ϵ, is also important in solving some problems. Consider for example that agents 1 and 2 both want A or B, and a third agent that wants AB. Allowing agents 1 and 2 to bid for both A and B, but only receive one of A or B in the allocation, solves the coordination problem, in which agents 1 and 2 must be allocated different items to out-bid the third agent.

4.6.2 Optimality Result

We first state an optimality result for the variations on CombAuction with non-anonymous prices.

(CS1a)	Agents maximize utility: $S_i^* \neq \emptyset \;\Rightarrow\; S_i^* = \arg\max_{S_i} v_i(S_i) \Leftrightarrow p_i(S_i)$ *true in every round because of best-response*
(CS1b)	Agents not in allocation happy: $S_i^* = \emptyset \;\Rightarrow\; \max_{S_i} v_i(S_i) \Leftrightarrow p_i(S_i) \leq 0$ *true in final round because of termination condition*
(CS2)	Auctioneer maximizes revenue: $S^* = \arg\max_{(S_1,\ldots,S_I)} \sum p_i(S_i)$ *true in every round because of price-update rules*

Table 4.6: Proof outline for CombAuction

Theorem 4.6 (CombAuction optimality). *(Optimality)* CombAuction(d) *and* CombAuction(3) *compute an allocation with value within* $3\min\{|\mathcal{G}|, |\mathcal{I}|\}\epsilon$ *of optimal.*

for $|\mathcal{G}|$ items, $|\mathcal{I}|$ agents, and ϵ bid increment.

Corollary 4.1 CombAuction(d) *and* CombAuction(3) *compute the efficient allocation and competitive equilibrium prices for a small enough bid increment* ϵ*.*

Clearly as ϵ gets smaller than the smallest finite difference in agents' values for bundles this converges to the optimal solution. Table 4.6 provides a sketched proof, while a full proof is provided below.

Recall that within CombAuction it is assumed that agents provide best-response information in response to prices in each iteration. Corresponding statements of the efficiency of *i*Bundle, the auction interpretation of CombAuction, make assumptions about agent behavior explicit (see Chapter 5).

Algorithm CombAuction(2), without the safety check and without non-anonymous prices, is also provably optimal in the following (sufficient) conditions:

Theorem 4.7 (anonymous optimality). CombAuction(2) *is an optimal primal-dual algorithm for CAP with anonymous prices in the following special-cases:*
(a) agents have additive or superadditive values, i.e. $v(S \cup S') \geq v(S) + v(S')$ for non-conflicting bundles S and S'; (b) agents demand bundles from the same partition of items,

e.g. all bids are for pairs of matching shoes, or single items; (c) the demand set of bundles for agent i, the bundles it bids for over the auction, is disjoint from the demand set of agent j, for all agents $i \neq j$; (d) bids from each agent are always overlapping; (e) bids from each agent are always for a single bundle.

A proof of conditions (a–c) follow quite easily from the proof of the optimality of CombAuction(d). When these conditions hold the auctioneer is sure to maximize revenue from bids in the next round of the auction without introducing non-anonymous prices, even in the case that bids from agents break the "safety" condition.[3] In case (b) the auction reduces to a simultaneous ascending-price auction on bundles in a fixed partition of items. The assignment problem, in which agents have unit-demand for items, is a special case of (b).

Conditions (d–e) follow trivially from the main result, because the safety condition holds in all rounds under these conditions. Single-minded bidders [LOS99] satisfy (e). Bidders that demand a core set of items with a selection from an additional set of items satisfy (d); consider for example a bidder in the FCC spectrum auction that needs New York, and then would like as many of the geographically neighboring licenses as possible.

4.6.3 Proof: CombAuction(2)

The optimality proof of CombAuction is inspired by a proof due to Bertsekas [Ber87] for Auction, an iterative primal-dual algorithm with an auction interpretation for the assignment problem.

I first prove optimality for CombAuction(2), in the special-case that the best-response bid sets are safe in all rounds of the auction. The proof of optimality for CombAuction(d) and CombAuction(3) follows from an equivalence between CombAuction(2) with a dummy item introduced for each agent and appended to its bids, and CombAuction(3).

In outline, I show that CombAuction implements a primal-dual algorithm for [LP_2] and [DLP_2], and computes integral solutions to [LP_2] when agents follow myopic best-response bidding strategies and bids are safe.

[3] For example, in the case of superadditive values whenever an agent bids for a compatible pair of bundles S and S' that violate the safety condition it must be the case that $p(S \cup S') > p(S) + P(S')$. This condition is sufficient to show that the auctioneer can continue to maintain (CS2) and maximize revenue from agent bids in the next round.

First, I show that the allocation and prices in each round of the auction correspond to feasible primal and dual solutions. Then, I show that the primal and dual solutions satisfy complementary-slackness conditions when the auction terminates.

In a particular round, let $\hat{S}_i$ denote the provisional allocation to agent i, and $p_{\text{ask}}(S)$ denote the ask price for bundle S.

Feasible primal. To construct a feasible primal solution assign $x_i(\hat{S}_i) = 1$ and $x_i(S') = 0$ for all $S' \neq \hat{S}_i$. Partition $y(k^*) = 1$ for $k^* = [\hat{S}_1, \ldots, \hat{S}_{|\mathcal{I}|}]$, and $y(k) = 0$ otherwise.

Feasible dual. To construct a feasible dual solution let dual variable $p(S)$ equal the ask price, $p_{\text{ask}}(S)$, on bundle S in CombAuction. The following values for $p(i)$ and π then satisfy constraints (DLP-1) and (DLP-2):

$$p(i) = \max\left\{0,\ \max_{S \subseteq G}\{v_i(S) \Leftrightarrow p(S)\}\right\} \tag{4.1}$$

$$\pi = \max_{k \in K} \sum_{S \in k} p(S) \tag{4.2}$$

The value $p(i)$ can be interpreted as agent i's maximum utility at the ask prices, and π can be interpreted as the maximum revenue that the auctioneer can achieve at the ask prices (irrespective of the bids placed by agents).

It is not necessary to explicitly compute $p(i)$, instead we will prove that allocation $\hat{S}$ and prices $p_{\text{ask}}(S)$ correspond to primal and dual solutions that satisfy complementary slackness conditions when the auction terminates.[4]

Complementary-slackness conditions. The first primal CS condition, CS-1 is:

$$x_i(S) > 0 \Rightarrow p(i) + p(S) = v_i(S) \tag{CS-1}$$

Given (4.1) it states that all agents must only receive a bundle that maximizes utility at the current prices. CS-1 is maintained throughout the auction because bundles are only allocated according to bids from agents, and agents place best-response bids.

Based on the best-response bidding strategy, we have

$$v_i(\hat{S}_i) \Leftrightarrow p_i^+(S) + \epsilon \geq \max\left\{0, \max_{S'}(v_i(S') \Leftrightarrow p_i^+(S'))\right\}$$

[4]This is just as well because the values $v_i(S)$ remain private information to agents during in CombAuction, and best-response is the only mode of interaction with an agent.

for any bundle $\hat{S}_i$ allocated to agent i. Then, because $p_{\text{ask}}(S) \geq p_i^+(S) \geq p_{\text{ask}}(S) \Leftrightarrow \epsilon$, $p(S) = p_{\text{ask}}(S)$, and $x_i(S_i) = 1$ implies that agent i bid for bundle S_i, we have by case analysis that:

$$x_i(S) > 0 \quad \Rightarrow \quad v_i(S) \Leftrightarrow p(S) + 2\epsilon \geq \max\left\{0,\ \max_{S'}(v_i(S') \Leftrightarrow p(S'))\right\}$$

Finally, substituting for $p(i)$ from (4.1), we prove ϵ-CS-1:

$$x_i(S) > 0 \ \Rightarrow p(i) + p(S) \leq v_i(S) + 2\epsilon \qquad (\epsilon\text{-CS-1})$$

The second primal CS condition, CS-2, is:

$$y(k) > 0 \Rightarrow \pi \Leftrightarrow \sum_{S \in k} p(S) = 0 \qquad (\text{CS-2})$$

Given (4.2) it states that the allocation must maximize the auctioneer's revenue at prices $p(S)$, over all possible allocations and irrespective of bids received from agents.

We prove that the provisional allocation, computed to maximize revenue based on agents' bids, is sufficient to maintain CS-2 in all rounds.

The following definition is useful:

DEFINITION 4.13 [strict positive price] An ask price $p_{\text{ask}}(S)$ is strictly positive if the price is greater than the ask price for every bundle contained in S, i.e. $p_{\text{ask}}(S) > p_{\text{ask}}(S')$ for all $S' \subset S$.

To see understand how CS-2 can hold in all iterations, notice:

(i) all bundles with strictly-positive prices receive bids in every round.

> Agent i with one of the highest losing bid for bundle S in round t will continue to bid for bundle S in rounds $t+1$. Let $u_i^t(S)$ denote agent i's utility for bundle S in round t. Then, $u_i^{t+1}(S) = u_i^t(S) \Leftrightarrow \epsilon$ because the ask price for S increases by ϵ. Also, $u_i^t(S) \geq u_i^t(S')$ for all bundles S' for which agent i did not bid in round t. Hence, with $u_i^t(S') \geq u_i^{t+1}(S')$ because the price of S' can only increase in round $t+1$, we have $u_i^{t+1}(S) \geq u_i^{t+1}(S') \Leftrightarrow \epsilon$, and a bid for S' can never exclude a bid for S from agent i's best-response bids in round $t+1$. A similar argument can be made for the utility of bundles that did receive a bid from agent i in round t.

(ii) all bundles in revenue-maximizing allocations receive bids from different agents.

No single agent causes the price to increase to its current level on a pair of compatible bundles. This follows because price updates are due to safe bids from agents. It is clear that this cannot happen in a single round. Furthermore, it can be shown by *induction* across rounds that an agent with a myopic best-response bidding strategy cannot increase the price of compatible bundles over a sequence of rounds without submitting unsafe bids in a single round.

Taking (i) with (ii), we have:

$$\sum_{i \in \mathcal{I}} p_i^+(\hat{S}_i) = \max_{k \in K} \sum_{S_i \in k} p_i^+(S_i)$$

Removing personalized prices, because $p_{\text{ask}}(S) \geq p_i^+(S) \geq p_{\text{ask}}(S) \Leftrightarrow \epsilon$, we have:

$$\sum_{i \in \mathcal{I}} p_{\text{ask}}(\hat{S}_i) + \min\{|\mathcal{G}|, |\mathcal{I}|\}\epsilon \geq \max_{k \in K} \sum_{S_i \in k} p_{\text{ask}}(S_i)$$

otherwise the optimal solution to $\max_{k \in K} \sum_{S_i \in k} p_{\text{ask}}(S_i)$ is a better solution with personalized prices $p_i^+(S)$ than $\hat{S} = (\hat{S}_1, \ldots, \hat{S}_I)$.

Finally, substituing for π from (4.2), and because $y(k) > 0$ implies $k = k^*$, and with $p(S) = p_{\text{ask}}(S)$, the partition representing the provisional allocation, we prove ϵ-CS-2:

$$y(k) > 0 \Rightarrow \pi \Leftrightarrow \sum_{S \in k} p(S) \leq \min\{|\mathcal{G}|, |\mathcal{I}|\}\epsilon \qquad (\epsilon\text{-CS-2})$$

The first dual CS condition, (CS-3), is:

$$p(i) > 0 \Rightarrow \sum_{S \subseteq G} x_i(S) = 1 \qquad (\text{CS-3})$$

Given (4.1) it states that every agent with positive utility for some bundle at the current prices must receive a bundle in the allocation. (CS-3) is only satisfied during the auction for agents that receive bundles in the provisional allocation, but we prove (CS-3) for all agents when CombAuction terminates.

We immediately have (CS-3) for the agents that satisfy termination condition [T1], because the agents receive a bundle in the provisional allocation. Similarly, we immediately

have (CS-3) for the agents that do not submit a best-response bid, because that implies that they have negative utility for all bundles at the prices. Finally, for agents in termination condition [T2] that receive no bundle but submit the same bids in two successive rounds; these agents must bid at ϵ below the ask price and have values just below ask prices otherwise prices would increase and their bids would change.

Finally, the last pair of dual CS conditions, (CS-4) and (CS-5), are:

$$p(S) > 0 \Rightarrow \sum_{i \in \mathcal{I}} x_i(S) = \sum_{k \in K, S \in k} y(k) \qquad \text{(CS-4)}$$

$$\pi > 0 \Rightarrow \sum_{k \in K} y(k) = 1 \qquad \text{(CS-5)}$$

The assignment $y(k^*) = 1$ for the partition $k^* = [\hat{S}_1 \dots \hat{S}_{|\mathcal{I}|}]$ trivially satisfies the right-hand side of both conditions.

Termination. By contradiction, assume the auction never terminates. Informally, [T2] implies that one or more agents must submit different bids in successive rounds, but with myopic best-response bidding this implies that prices must increase and if the auction does not terminate than one or more agents must eventually bid above their values for bundles— a contradiction with myopic best-response.

Putting it all together.

Finally, we prove the worst-case error term in Theorem 4.6 when the auction terminates. Let $S^* = (S_1^*, \dots, S_{|\mathcal{I}|}^*)$ denote the final allocation.

Summing ϵ-CS-1 over all agents in the final allocation, and with $p(i) = 0$ for agents not in the allocation by (CS-3),

$$\sum_{i \in \mathcal{I}} p(i) \leq \sum_{i \in \mathcal{I}} v_i(S_i^*) \Leftrightarrow \sum_{i \in I} p(S_i^*) + 2 \min\{|\mathcal{G}|, |\mathcal{I}|\}\epsilon$$

because an allocation can include no more bundles than there are items or agents. Introducing ϵ-CS-2, because $y(k^*) = 1$ for final allocation S_i^*, then $\pi \leq \sum_{i \in \mathcal{I}} p(S_i^*) + \min\{|\mathcal{G}|, |\mathcal{I}|\}\epsilon$.

Adding these two equations, we have:

$$\pi + \sum_{i \in \mathcal{I}} p(i) \leq \sum_{i \in \mathcal{I}} v_i(S_i^*) + 3 \min\{|\mathcal{G}|, |\mathcal{I}|\}\epsilon$$

The left-hand side is the value of the final dual solution, V_{DLP}, and the first-term on the right-hand side is the value of the final primal solution, V_{LP}. We know $V^*_{\text{LP}} \leq V_{\text{DLP}}$, where V^*_{LP} is the value of the optimal primal solution, by the weak-duality property of linear programs.

Thus, because $V_{\text{DLP}} \leq V_{\text{LP}} + 3\min\{|\mathcal{G}|, |\mathcal{I}|\}\epsilon$, it follows that

$$V_{\text{LP}} \geq V^*_{\text{LP}} \Leftrightarrow 3\min\{|\mathcal{G}|, |\mathcal{I}|\}\epsilon$$

Finally, because the primal solution is integral (by construction during CombAuction), it is a feasible and optimal solution to the combinatorial resource allocation problem. ∎

4.6.4 Proof: CombAuction(d) and CombAuction(3)

A simple transformation of agents' bids reduces any problem in CombAuction(d) or CombAuction(3) to a safe problem in CombAuction(2). Bertsekas [Bet92] proposes a similar transformation method to derive an Auction method for combinatorial optimization problems such as max-flow and the transportation problem, reducing problems to the Assignment problem.

In CombAuction a simple transformation of agents' bids allows CombAuction(3) to be implemented within CombAuction(2) without price-discrimination, and ensures that agents' bids remain safe throughout the auction.

Whenever bids from agent i are not safe in CombAuction(2) we can simulate the price-update rule in CombAuction(3) by introducing a new dummy item that is specific to that agent, call it x_i. This item is concatenated by the auctioneer to all bids from agent i in this round and all future rounds. It has the following effects:

1. The outcome of winner-determination, or the allocative efficiency of the auction, is unchanged because no other agent bids for item x_i.

2. Agent i's bids are always safe because every bid includes item x_i, and no pair of bids is compatible.

3. The price increases due to bids from agent i are isolated to that agent in all future rounds because all price increases are for bundles that include item x_i.

The optimality of CombAuction(3) follows immediately from the optimality of CombAuction(2) without price discrimination.

4.7 Earlier Primal-Dual Auction Methods

Prior to *i*Bundle there was no method to terminate in competitive equilibrium in the general combinatorial allocation problem (CAP). Table 4.7 summarizes the progress in iterative auction design over the past two decades. Each contribution relaxes assumptions on agent preferences and/or strengthens the equilibrium analysis of the auction. Bidding languages differ in terms of whether agents can bid on items or bundles, whether agents can submit single or multiple bids, and the logic used to combine multiple bids. Prices differ in terms of whether individual items or bundles are priced, and whether prices are anonymous or non-anonymous.

Name	Assumptions	Price structure	Bid structure	Update method	outcome
CK81	linear-additive	items	OR-items	greedy	CE
KC82	GS	items	one bundle	greedy	CE
Ber79–87	unit-demand	items	one item	greedy	CE
DGS86	unit-demand	items	XOR-items	minimal	Vickrey
Aus97	homog subadditive	one price	one bundle	clinch	Vickrey
Aus00	GS	items	one bundle	clinch & unclinch $n+1$ auctions	Vickrey
GS00	GS	items	XOR-bundle	minimal	min CE
Wur00	monotone	bundles	XOR-bundle	minimal	-
iBundle(2)	safe	bundles	XOR-bundle	greedy	CE
iBundle(d)	monotone	bundles non-anonymous	XOR-bundle	greedy	CE
Adjust	monotone	bundles non-anonymous	XOR-bundle	greedy	min CE
Extend&Adjust	monotone	bundles non-anonymous	XOR-bundle	greedy	(Vickrey)

Table 4.7: Primal-dual auction methods.

All auctions terminate with efficient allocations, but achieve different levels of robustness-to-manipulation. Vickrey payments provide more robustness-to-manipulation than min CE prices, which provide more robustness than CE prices. Termination with Vickrey payments makes myopic best-response a Nash equilibrium of the auction [GS00]. Termination with competitive equilibrium prices provides some incentive-compatibility, at

least in the last round of the auction. Minimal CE prices often coincide with Vickrey payments, but otherwise might be imagined to provide "intermediate" incentive-compatibility properties between that of any CE outcome and the Vickrey outcome.

Computing adjusted prices after termination with *i*Bundle and ADJUST (see Chapter 7), labeled (Adjust) in Table 4.7, implements *minimal* CE prices. These prices support Vickrey payments for both unit-demand and gross-substitutes agent preferences, i.e. capturing all known previous iterative Vickrey auctions for the CAP problem. Moreover, I believe that the extended auction, *i*Bundle Extend&Adjust, is significantly more powerful. I conjecture that the extended auction is an iterative Vickrey auction for *all* CAP instances. The relationship between the final prices in an auction and an auction's robustness to manipulation is discussed in detail in Chapters 6 and 7.

The main *structural differences* across auctions are in the *bidding-languages*, the *prices*, and the price-update rules. Beyond that, with the exception of Ausubel [Aus97, Aus00], the auctions share the following essential steps:

(a) announce prices and a provisional allocation

(b) receive *myopic best-response* bids from agents

(c) look for a provisional allocation that satisfies every agent

(d) increase prices on over-demanded items (or bundles) if no solution is found

The auctions differ in the method used to adjust prices across rounds, i.e. the choice of overdemanded set. Price update methods are either *greedy* or *minimal*. In a greedy update rule the price is increased on all over-demanded items (or bundles). In a minimal update rule the price is increased on the smallest set of over-demanded items (or bundles). Prior to the *i*Bundle Extend&Adjust method, the usual approach to compute Vickrey payments was to implement minimal price-updates in each round [DGS86, GS00] of the auction. Minimal price updates are designed to adjust towards *minimal* CE prices, which are equivalent to Vickrey payments in many problems [GS99]. The dual solution that maximizes agent utility and minimizes auctioneer revenue corresponds to the minimal competitive equilibrium solution. In *i*Bundle Extend&Adjust prices are increased beyond minimal CE prices, but adjusted back towards minimal CE prices, and Vickrey payments, after termination.

4.7.1 Assumptions on Agent Preferences

The methods of Bertsekas [Ber79, Ber81, Ber88] and Demange *et al.* [DGS86] assume *unit-demand* preferences, in which each agent demands at most one item:

DEFINITION 4.14 [unit-demand] Agent i's valuation

$$v_i(S) = \max_{j \in S} v_i(j)$$

The allocation problem for unit-demand preferences is known as the *assignment problem*, with the goal to assign a single item to each agent to maximize value.

Crawford & Knoer [CK81] assume linear-additive agent preferences.

DEFINITION 4.15 [linear-additive] Agent i's valuation

$$v_i(S) = \sum_{j \in S} v_i(j)$$

Kelso & Crawford [KC82], Gul & Stacchetti [GS00], and Ausubel [Aus00] assume that agents have *gross-substitutes* (GS) preferences. Gross-substitutes states that if the prices were increased from p to p' then the agent would continue to demand any item whose price did not increase (see Definition 4.8). Unit-demand preferences and linear-additive preferences are special cases of GS.

Ausubel's [Aus97] ascending-price auction assumes multiple identical (homogeneous) items and subadditive preferences:

DEFINITION 4.16 [subadditive] Valuation function $v_i(m)$, for $m \geq 0$ units of an item, is subadditive if the marginal value for each additional item is (weakly) decreasing, i.e. if $v_i(n+2) \Leftrightarrow v_i(n+1) \;\geq\; v_i(n+1) \Leftrightarrow v_i(n)$ for $n \geq 0$ units.

Finally, in *i*Bundle, and in A*k*BA [WW00], the only restriction on agent preferences is that they are *monotone*:

DEFINITION 4.17 [monotone] Valuation function $v_i(S)$ is monotone if $v_i(S_1) \geq v_i(S_2)$ for all $S_1 \supseteq S_2$, i.e. free-disposal of additional items.

4.7.2 Relating to Primal-Dual Methods

Bertsekas [Ber79, Ber81, Ber88] was the first to make an explicit connection between primal-dual algorithms and auction mechanisms for combinatorial optimization problems. AUCTION is a primal-dual based method to solve the assignment problem, in which agents have unit-demand preferences. Although AUCTION has a natural auction interpretation, with agents bidding for their favorite item in each round, there is no consideration of agent incentives in Bertsekas' work. His motivation was to develop new algorithms to solve problems in parallel environments, although experimental analysis showed good performance on single processors. See [Ber87] for a text book introduction to the AUCTION algorithm, and [Ber90] for a tutorial.

Bertsekas has also proposed variants of his AUCTION algorithm for other combinatorial optimization problems, including the *transportation problem* [BC89], a variation of assignment with multiple identical items, and the minimal cost flow problem [Ber86]. Bertsekas [Bet92] has recently provided a unified framework of this large body of work, transforming each optimization problem into the assignment problem.

The work of Demange *et al.* [DGS86] is important, as it was the first to consider agent incentives and demonstrate an iterative auction procedure to compute Vickrey payments in the assignment problem. The *DGS* procedure was derived in the context of a linear program formulation for Vickrey payments due to Leonard [Leo83]. Sankaran [San94] appears to provide a primal-dual interpretation of DGS. Recently, Bikchandani *et al.* [BdVSV01] provide an alternative primal-dual interpretation of DGS, with respect to a new linear program formulation of the CAP.

Although primal-dual analysis is not explicit in the work of [CK81, KC82, DGS86, Aus97, Aus00, GS00] a primal-dual explanation can be provided for an appropriate linear programming model [BdVSV01].

Ausubel's auctions [Aus97, Aus00] are quite innovative. In Ausubel [Aus97] the auction maintains one explicit price, but is able to compute multiple prices, one for each agent, that equal the Vickrey payments. Ausubel describes a "clinching" process, whereby the price for items is locked-in during the course of the auction. Bikchandani & Ostroy [BO00] give a primal-dual algorithm and linear program for Ausubel's [Aus97] homogeneous good auction. Bikchandani *et al.* [BdVSV01] propose an alternative derivation based on a network path planning formulation. The clinching rule is reinterpreted as a discount from

a final clearing price in the auction.

Recently Ausubel [Aus00] proposes a complex auction procedure to compute Vickrey payments with GS agent preferences. Ausubel maintains one price for each item, i.e. linear prices, but actually must run $I+1$ auctions (a main auction, and one without each agent in turn) to compute Vickrey payments. The auction's theoretical contribution is significant because it has been observed [GS00] that no single set of linear competitive equilibrium prices can support Vickrey payments with GS agent preferences. Note however that there is a set of non-linear (and perhaps non-anonymous) competitive equilibrium prices that support the Vickrey payments with GS preferences [BdVSV01].

Gul & Stacchetti [GS00] state that *no* dynamic mechanism can reveal sufficient information to implement the Vickrey mechanism with GS preferences. Ausubel's method is outside the spirit of their negative result because of its reliance on multiple auctions. Similarly, my work on *i*Bundle escapes this negative result with non-linear and non-anonymous prices on items. The final price adjust step might also be outside of the spirit of their focus on "monotonic" price adjustment. *i*Bundle with ADJUST will provably compute the efficient allocation and Vickrey payments for GS agent preferences, based on myopic best-response information from agents to an ascending sequence of prices.

Wurman's Ascending k-Bundle Auction (AkBA) [WW00] family of iterative combinatorial auctions also maintain explicit prices on bundles of items, and were designed with a generalization of the DGS [DGS86] auction in mind. In each round of AkBA Wurman uses a linear program to increase (and sometimes decrease) prices. A1BA, thought to be the most promising of the family, computes the *maximal* prices that solve a restricted dual problem in each round.

Chapter 5

*i*Bundle: An Iterative Combinatorial Auction

In this chapter I introduce the *i*Bundle ascending-price combinatorial auction, which follows quite directly from the primal-dual algorithm CombAuction for the combinatorial allocation problem. The best-response information provided by agents in CombAuction has a natural interpretation as a utility-maximizing bidding strategy for a myopic agent, i.e. an agent that takes the current prices as fixed and does not look beyond the current round.

Even without the added incentive properties that are inherited from the Vickrey-Clarke-Groves mechanism via Extend&Adjust (see Chapter 7) the contribution of *i*Bundle is significant. It is the first iterative auction to provably terminate with an efficient allocation for a reasonable agent bidding strategy, without any restrictions on agents' valuation functions. We make no assumptions about agent valuation functions, other than monotonicity (or free-disposal). The main design decisions in *i*Bundle are:

- Exclusive-or bids over bundles of items.
- A simple price-update rule with minimal consistency requirements on prices across different bundles.
- A dynamic method to determine when non-anonymous prices are required to price the efficient allocation in competitive equilibrium.

Myopic best-response need not be an agent's optimal sequential strategy in *i*Bundle, and the basic auction design is *not* strategy-proof like the GVA. This is addressed in Chapter 7 with Extend&Adjust, a method to keep *i*Bundle open for a second phase and

compute Vickrey payments.

The basic auction procedure is described for an exclusive-or bidding language over items. However, as described in Section 5.6, there are natural extensions to more expressive languages for particular domains

Experimental results confirm the efficiency of *i*Bundle across a set of combinatorial allocation problems from the literature. The auction computes efficient solutions, even with quite large bid increments. Results also demonstrate that non-anonymous prices are only important above 99% allocative efficiency. Information revelation in *i*Bundle is measured for a simple metric, that considers the degree to which agents reveal their complete valuation function via their bids during the auction. *i*Bundle is shown to have *scalable* performance, only requiring agents to reveal a small amount of information in easy problems.

The auctioneer in *i*Bundle solves one winner-determination problem in each round, compared to one for each agent in the final allocation in the GVA. Although the problem of computing a provisional allocation in each round remains NP-hard the problem instances in *i*Bundle are much smaller than in the GVA because the agents only bid for a small subset of bundles in each round. In addition, a number of tricks allow speed-ups:

(a) we can use cached solutions across rounds to speed-up solving a sequence of related winner-determination problems;

(b) we can make trade-offs between allocative efficiency and winner-determination time by adjusting the speed with which prices are increased in the auction;

(c) we can introduce approximation algorithms with a simple *bid monotonicity property*, and still maintain incentives for the same myopic bidding strategy.

Computational results demonstrate an order-of-magnitude speed-up over the VCG mechanism at 99% allocative efficiency, with the same combinatorial optimization algorithm to solve winner-determination problems in both mechanisms. An approximate winner-determination algorithm also proves useful. *i*Bundle can often achieve greater than 90% efficiency with negligible computation using a simple greedy algorithm in each round.

The outline of this chapter is as follows. The first section gives a full description of *i*Bundle, including the bidding language, price update rules, and winner-determination rules. Section 5.2 states the main theoretical results, which follow from the optimality

proofs for CombAuction. Section 5.3 introduces the experimental methods: agent models, problem sets, winner-determination algorithm, etc. The experimental results are split over two sections. Section 5.4 is concerned with the efficiency and information revelation properties of *i*Bundle, and based around Parkes [Par99]. Section 5.5 is concerned with the winner-determination complexity and communication properties, and based around Parkes & Ungar [PU00a]. Section 5.6 outlines special cases of *i*Bundle for restricted bidding languages. Finally, Section 5.7 compares *i*Bundle with earlier iterative combinatorial auction designs.

5.1 Auction Description

*i*Bundle has three variations, *i*Bundle(2), *i*Bundle(d) and *i*Bundle(3), which differ in their price update rules.[1] The *i*Bundle(d) variation introduces price discrimination dynamically, only as required to support the efficient allocation in competitive equilibrium. It has provable allocative efficiency for myopic best-response strategies. Each variation implements the associated CombAuction variation when agents follow myopic best-response strategies. The rules make *i*Bundle as robust as possible against non-myopic agent strategies. For example, one rule states that an agent must repeat its bid (at the same or greater price) in the next round for any bundle it receives in the provisional allocation. This rule only restricts non-myopic strategies because an agent with a myopic best-response strategy would always want to repeat its bid for a bundle that does not increase in price, because the prices only increase on other bundles and never decrease.

Recall that $\mathcal{G}$ denotes the set of items to be auctioned, $\mathcal{I}$ denote the set of agents, and $S \subseteq \mathcal{G}$ denote a bundle of items. The auction proceeds in rounds, indexed $t \geq 1$. We describe the types of bids that agents can place, and the allocations and price updates computed by the auctioneer.

Bids. Agents can place exclusive-or bids for bundles, e.g. S_1 XOR S_2, to indicate than an agent wants either all items in S_1 or all items in S_2 but not both S_1 and S_2. Agent i associates a *bid price* $p^t_{\text{bid},i}(S)$ with a bid for bundle S in round t, non-negative by definition. The price must either be within ϵ of, or greater than, the *ask price* announced by the auctioneer (see below). Parameter $\epsilon > 0$ defines the *minimal bid increment*, the

[1] I will sometimes use *i*Bundle both to refer to the family of auctions in general, and *i*Bundle(d) in particular.

minimal price increase in the auction. Agents must repeat bids for bundles in the current allocation, but can bid at the same price if the ask price has increased since the previous round. An agent can also bid ϵ below the ask price for any bundle in any round— but then it cannot bid a higher price for that bundle in the future. This allows an agent to bid for a bundle priced slightly above its value.

Winner-determination. The auctioneer solves a winner-determination problem in each round, computing an allocation of bundles to agents that maximizes revenue. The auctioneer must respect agents' XOR bid constraints, and cannot allocate any item to more than one agent. The provisional allocation becomes the final allocation when the auction terminates. Ties are broken in favor of assigning bundles to more agents, and then at random, except when the same bids are received in two successive rounds when the auctioneer selects the same allocation (and the auction terminates).

Prices. The price-update rule generalizes the rule in the English auction, which is an ascending-price auction for a single item. In the English auction the price is increased whenever two or more agents bid for the item at the current price. In *i*Bundle the price on a bundle is increased when one or more agents that do not receive a bundle in the current allocation bid at (or above) the current ask price for a bundle. The price is increased to ϵ (the minimal bid increment) above the greatest failed bid price. The initial ask prices are zero.

The auctioneer announces a new ask price, $p^t_{\text{ask}}(S)$ in round t, for all bundles S that increase in price. Other bundles are implicitly priced at least as high as the greatest price of any bundle they contain, i.e. $p_{\text{ask}}(S') \geq p_{\text{ask}}(S)$ for $S' \supseteq S$. These ask prices are anonymous, the same for all agents.

Price discrimination. In some problems the auctioneer introduces price discrimination based on agents' bids, with different ask prices to different agents, when this is necessary to achieve an optimal allocation. A simple rule dynamically introduces price-discrimination (or non-anonymous prices) on an agent-by-agent basis, when an agent submits bids that are not *safe*. As in CombAuction, safe bids are defined as:

Definition 5.1 [safe bids] An agent's bids are safe if the agent is allocated a bundle in the current allocation, or it does not bid at or above the ask price for any pair of compatible bundles S_1, S_2, such that $S_1 \cap S_2 = \emptyset$.

When an agent's bids are not safe the agent receives *individual ask prices*, $p_{\text{ask},i}(S)$, in future rounds. Individual prices are initialized to the current general prices, $p^t_{\text{ask},i}(S) = p^t_{\text{ask}}(S)$, and increased to ϵ above the agent's bids in future rounds that the agent receives no bundle in the provisional allocation. Intuitively, even though bids for compatible bundles S_1 and S_2 from a single agent i are unsuccessful, it remains possible that bids for the bundles from two *different* agents can succeed at the same price because the XOR bid constraint prevents the auctioneer accepting multiple bids from agent i.

Termination. The auction terminates when every agent that bids either [T1] receives a bundle in the provisional allocation, or [T2] repeats the same bids in two consecutive rounds.

5.1.1 A Myopic Best-Response Bidding Strategy

*i*Bundle computes an optimal allocation with *myopically rational* agents that play a best (utility-maximizing) response to the current ask prices and allocation in the auction. The agents are myopic in the sense that they only consider the current round of the auction.

Let $v_i(S)$ denote agent i's value for bundle S, and assume $v_i(\emptyset) = 0$ and free disposal of items, so that $v_i(S') \geq v_i(S)$ for all $S' \supseteq S$. Consider a risk-neutral agent, with a quasi-linear utility function $u_i(S) = v_i(S) \Leftrightarrow p(S)$ for bundle S at price $p(S)$. Further, assume that agents are indifferent to within a utility of $\pm\epsilon$, the minimal bid increment. This is reasonable as $\epsilon \to 0$.

By definition, a myopic agent bids to maximize utility at the current ask prices (taking an ϵ discount when repeating a bid for a bundle in the provisional allocation or bidding for a bundle priced just above its value). The myopic best-response strategy is to submit an XOR bid for all bundles S that maximize (to within ϵ) utility $u_i(S)$ at the current prices. This maximizes the probability of a successful bid for bid-monotonic WD algorithms.

5.1.2 Discussion

Unlike earlier auction designs for the combinatorial allocation problem, *i*Bundle permits both non-linear prices (i.e. prices on bundles not equal to the sum of prices on items) and non-anonymous prices (i.e. different price for the same bundle to different agents). Given that competitive equilibrium is a central solution concept in efficient auction design, and that both non-linear and non-anonymous prices are required for competitive equilibrium

in some CAP problem instances, this appears essential to the efficiency of *i*Bundle [BO99]. *i*Bundle is the first iterative auction to terminate with the efficient allocation in the general CAP problem for a reasonable agent bidding strategy, in this case myopic best-response. *i*Bundle also terminates in competitive equilibrium, such that the allocation simultaneously maximizes the auctioneer's revenue and every agent's utility at the final prices.

The other main feature of the price-update rules in *i*Bundle is that only weak consistency is enforced across prices. Prices may be subadditive *or* superadditive, the only requirement is that they satisfy a simple monotonicity requirement:

$$p_i(S_1) \geq p_i(S_2) \qquad \text{, if } S_1 \supseteq S_2$$

for bundles S_1 and S_2. Additional consistency rules fail to characterize competitive equilibrium prices in some CAP problem instances.

Allowing agents to bid for bundles of items avoids the *exposure problem* identified in [BCL00] for auctions that do not allow combinatorial bids, for example simultaneous ascending-price auctions. The exposure problem occurs when an agent loses its bid on one-or-more items in a bundle and is left with an incomplete bundle. Bundle bids allow agents to make explicit statements about contingencies, for example a bid on bundle AB states "I only want A if I also get B".

Exclusive-or bids are not compact representations of an agent's demand in some problems, for example when an agent has a linear-additive valuation function (see Section 3.2.2). We can derive price-update rules for other bid languages [Par99]. For example, the auctioneer can convert OR bids into equivalent XOR bids by creating a new "dummy" agent to submit an XOR bid for each bundle that receives an OR bid from an agent (see Section 5.6).

Non-linear prices can require enforcement, for example to prevent the possibility of arbitrage in which a third-party profits from subadditive prices on bundles ($p(S_1 \cup S_2) < p(S_1) + p(S_2)$) by purchasing bundles to be "disassembled" and sold for profit. Similarly, with subadditive prices a bidding cartel can form to take advantage of bundle discounts, and this can also distort the efficiency of the mechanism. A single agent might try to avoid superadditive prices, with $p(S_1 \cup S_2) > p(S_1) + p(S_2)$, by entering the auction under multiple pseudonyms and purchasing smaller bundles for "assembly" after purchase.

In addition, the variations of *i*Bundle with discriminatory prices, e.g. $p_i(S) \neq p_j(S)$ for agents i, j, may require an auctioneer to prevent agents entering under multiple pseudonyms.

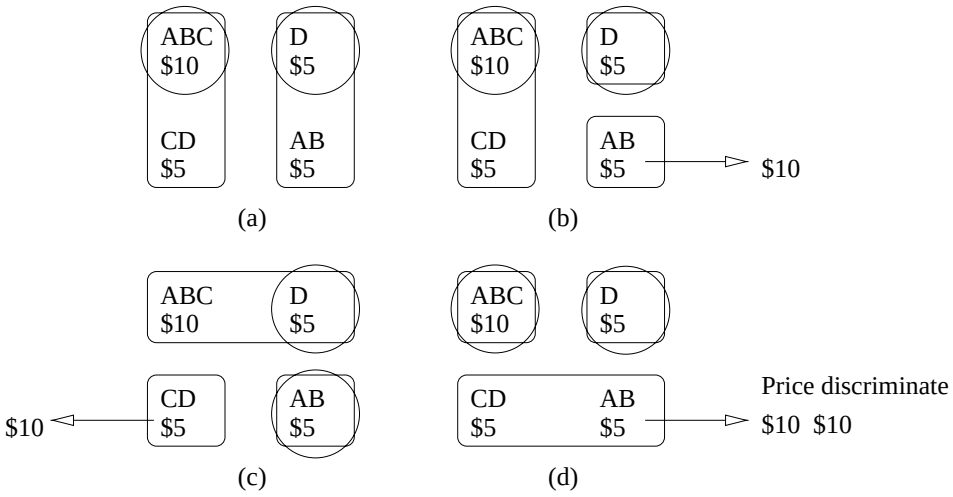


Figure 5.1: Auction scenarios.

Methods of enforcement include: prevent the transfer of items between agents in an aftermarket (*e.g.* the airline industry); and prevent agents from entering an auction under multiple pseudonyms, for example through cryptographic message authentication and signature techniques.

5.1.3 Example Auction Scenarios

Figure 5.1 shows four different auction scenarios that illustrate winner-determination and price-updates. In each scenario, bundles ABC, CD, D and AB each receive a bid from some agent, but the scenarios differ in the agents that submit the bids. The 'boxes' indicate XOR bids from the same agent, and the 'circles' indicate the allocation selected by the auctioneer to maximize revenue. Price increases are indicated with an 'arrow'. The minimal bid increment $\epsilon = 5$. Notice that the bid prices for bundles are consistent, such that $p(ABC) \geq p(AB)$ and $p(CD) \geq p(D)$. This must be maintained in *i*Bundle.

The auctioneer selects the same revenue-maximizing allocation in scenarios (a), (b) and (d), allocating bundles ABC and D to two different agents. In (c) the same agent bids for bundle ABC and D and the auctioneer must select another allocation, because it must respect the XOR bid constraint. Allocation D and AB is chosen in preference to ABC because it includes more agents.

In (a) the auction terminates because the revenue-maximizing allocation includes a bid from every agent that bids. In (b) the auction does not terminate because the agent that

bids $(AB, \$5)$ is not happy. The ask price for AB is increased to $5 + \epsilon$, the minimal bid increment, in the next round. Scenario (c) is similar, except that the provisional allocation is different, and the price is increased on CD in the next round. In (d) the bids from the agent that receives no bundle in the provisional allocation are not *safe* because bundle CD and AB are compatible. The auctioneer introduces individual prices for this agent in the next round. The ask price remains unchanged for the agents in the provisional allocation, but ask prices increase to $5 + \epsilon$ for bundles CD and AB to the third agent.

5.1.4 Worked Examples

Consider Problem 4 in Table 5.1, in which agent 1 wants *either* A or B, and has value 2 for each, and agent 2 wants only A and B, and has value 3 for AB. The efficient allocation is to allocate AB to agent 1. Table 5.2 illustrates the ask prices, allocation, and bids from agents in each round of *i*Bundle(*2*). The bid increment is $\epsilon = 0.5$. Notice that agent 1 bids ϵ below the ask price in round 4 because it repeats a bid for a bundle in the current allocation, and agent 2 takes a discount in round 11 because the ask prices are greater than its values. The provisional allocation in each round is indicated $*$. The auction terminates with the optimal allocation $[1, AB]$ in round 11 because it receives the same bids as in round 10. The final prices are subadditive, in fact as $\epsilon \to 0$ the final prices approach $p(A) = p(B) = 2 = p(AB)$. The ask price on AB remains within ϵ of the *maximum* (and not the sum) of the prices on A and B because agent 2 bids for both A and B and the auctioneer cannot sell multiple bundles to the same agent.

	A	B	AB
Agent 1	0	0	3^*
Agent 2	2	2	2

Table 5.1: Problem 4

There are no linear (i.e. prices on items) that support the efficient allocation in Problem 4 in competitive equilibrium. Actually, there are not even any non-linear and anonymous prices that support the efficient allocation in competitive equilibrium. For examples, prices $p(AB) = p(A) = p(B) = 2.1$ are not in competitive equilibrium with allocation $[1, AB]$ because the auctioneer wants to sell A and B separately to maximize revenue. Non-linear and non-anonymous prices, such as $p_1(A) = p_1(B) = 0, p_1(AB) = 2.5$ and $p_2(A) = p_2(B) = p_2(AB) = 2.1$ are required to support the efficient allocation in competitive equilibrium.

Round	Prices			Bids				Allocation	Revenue
	A	B	AB	Agent 1	Agent 2				
1	0	0	0	(AB, 0)*	(A, 0)	(B, 0)	(AB, 0)	[1, AB]	0
2	0.5	0.5	0.5	(AB, 0)	(A, 0.5)*	(B, 0.5)	(AB, 0.5)	[2, A]	0.5
3	0.5	0.5	0.5	(AB, 0.5)*	(A, 0.5)	(B, 0.5)	(AB, 0.5)	[2, A]	0.5
4	1	1	1	(AB, 0.5)	(A, 1)	(B, 1)*	(AB, 1)	[2, B]	1
5	1	1	1	(AB, 1)	(A, 1)	(B, 1)*	(AB, 1)	[2, B]	1
6	1	1	1.5	(AB, 1.5)*	(A, 1)	(B, 1)	(AB, 1.5)	[1, AB]	1.5
7	1.5	1.5	2	(AB, 1.5)	(A, 1.5)	(B, 1.5)	(AB, 2)*	[2, AB]	2
8	1.5	1.5	2	(AB, 2)*	(A, 1.5)	(B, 1.5)	(AB, 2)	[1, AB]	2
9	2	2	2.5	(AB, 2)	(A, 2)*	(B, 2)	(AB, 2)	[2, A]	2
10	2	2	2.5	(AB, 2.5)*	(A, 2)	(B, 2)	(AB, 2)	[1, AB]	2.5
11	2.5	2.5	2.5	(AB, 2.5)*	(A, 2)	(B, 2)	(AB, 2)	`terminates.`	

Table 5.2: *i*Bundle(2) on Problem 4, Bid incr. $\epsilon = 0.5$. Provisional allocations indicated *.

It is interesting that *i*Bundle(2) terminates with the efficient allocation in Problem 4 even without non-anonymous prices. In fact, *safety* fails in many rounds, for example in rounds 3, 6, 8 and 10, and *i*Bundle(d) would introduce a separate set of prices for agent 2. The anonymous prices combined with the winner-determination rule in each round of *i*Bundle(2), which prevents the auctioneer selling A and B to agent 2, simulate the effect of non-anonymous competitive equilibrium prices— essentially the allocation $[1, AB]$ does become the revenue-maximizing solution for the auctioneer because it is unable to sell both A and B simultaneously.

	A	B	AB
Agent 1	0	0	3
Agent 2	2*	0	2
Agent 3	0	2*	2

Table 5.3: Problem 5

Let us now consider Problem 5 in Table 5.3, in which the efficient allocation is to allocate A to agent 2 and B to agent 3. Linear prices *are* sufficient to support the optimal allocation in competitive equilibrium in this problem, for example $p(AB) = p(A) + p(B) = 1.6 + 1.6 = 3.2$. Table 5.4 illustrates the progress of *i*Bundle(2). The final prices in this case are approximately linear, and as $\epsilon \to 0$, the ask prices approach $p(A) = p(B) = 1.5$ and $p(AB) = 3$. In this case a successful bid for AB must be at least the *sum* (not the maximum) of the bid prices for A and B, because different agents— agents 2 and 3 —bid for A and B. This explains why the ask price for AB remains within ϵ of the sum of the

Round	Prices			Bids						Allocation	Revenue
	A	B	AB	Agent 1	Agent 2		Agent 3				
1	0	0	0	(AB, 0)	(A, 0)*	(AB, 0)	(B, 0)*	(AB, 0)		[2, A], [3, B]	0
2	0	0	0.5	(AB, 0.5)*	(A, 0)	(AB, 0.5)	(B, 0)	(AB, 0.5)		[1, AB]	0.5
3	0.5	0.5	1	(AB, 0.5)	(A, 0.5)*	(AB, 1)	(B, 0.5)*	(AB, 1)		[2, A], [3, B]	1
4	0.5	0.5	1	(AB, 1)	(A, 0.5)*	(AB, 1)	(B, 0.5)*	(AB, 1)		[2, A], [3, B]	1
5	0.5	0.5	1.5	(AB, 1.5)*	(A, 0.5)		(B, 0.5)			[1, AB]	1.5
6	1	1	1.5	(AB, 1.5)	(A, 1)*	(AB, 1.5)	(B, 1)*	(AB, 1.5)		[2, A], [3, B]	2
7	1	1	2	(AB, 2)	(A, 1)*		(B, 1)*			[2, A], [3, B]	2
8	1	1	2.5	(AB, 2.5)*	(A, 1)		(B, 1)			[1, AB]	2.5
9	1.5	1.5	2.5	(AB, 2.5)	(A, 1.5)*	(AB, 2)	(B, 1.5)*	(AB, 2)		[2, A], [3, B]	3
10	1.5	1.5	3	(AB, 3)	(A, 1.5)*		(B, 1.5)*			[2, A], [3, B]	3
11	1.5	1.5	3.5	(AB, 3)	(A, 1.5)*		(B, 1.5)*			`terminates.`	

Table 5.4: *i*Bundle(2) on Problem 5, Bid incr. $\epsilon = 0.5$. Provisional allocations indicated *.

ask prices for A and B throughout the auction.

Notice that bid safety is never violated by the bids from an unsuccessful agent in any round of *i*Bundle in Problem 5, providing direct evidence that non-anonymous prices are not required for a competitive equilibrium outcome.

5.2 Theoretical Results

We can make an immediate claim about the efficiency of *i*Bundle with agents that follow myopic best-response strategies, based on the analysis of the primal-dual algorithm CombAuction in the previous chapter. See Parkes & Ungar [PU00a] for a direct proof.

Recall that $|G|$ is the number of items, $|I|$ is the number of agents, and ϵ is the minimal bid increment.

Theorem 5.1 (optimality). *iBundle terminates with an allocation that is within* $3\min\{|G|, |I|\}\epsilon$ *of the optimal solution, for best-response agent bidding strategies.*

The auction is *optimal* as the bid increment approaches zero because the error-term goes to zero.

As described earlier, the optimality follows from a primal-dual analysis: the provisional allocation is always a feasible primal solution, the prices a feasible dual solution, and complementary-slackness conditions hold on termination.

In a simpler variation, *i*Bundle(2), the auctioneer never tests for bid-safety and never introduces price discrimination.

THEOREM 5.2 (anonymous optimality). i*Bundle(2) terminates with an allocation that is within* $3\min\{|G|, |I|\}\epsilon$ *of the optimal solution when bids are* safe, *for best-response agent bidding strategies.*

As noted in the context of COMBAUCTION (Theorem 4.7), special cases on agent preferences for which *i*Bundle(2) remains efficient, include the following:
(1) Every agent demands different bundles
(2) Agents have additive or superadditive values, i.e. $v(S \cup S') \geq v(S) + v(S')$ for non-conflicting bundles S and S'
(3) The bundles that receive bids throughout the auction are from a single partition of items, e.g. bids are for pairs of matching shoes or individual items.

5.3 Experimental Methods

Experimental results support the theoretical claims about the allocative efficiency of *i*Bundle. Results also demonstrate that *i*Bundle continues to perform well with quite large bid increments, and without price discrimination. The performance of *i*Bundle is tested on randomly generated problems sets and with myopic best-response agent bidding strategies.

5.3.1 Metrics

Given allocation $\mathbf{S} = (S_1, \ldots, S_I)$ we compute the following metrics:

[**Efficiency**] Allocative efficiency, $\mathit{eff}(\mathbf{S})$, is measured as the ratio of the value of the final allocation S to the value V^* of the optimal allocation that maximizes total value across the agents: $\mathit{eff}(\mathbf{S}) = \sum_i v_i(S_i)/V^*$ where $v_i(S_i)$ is agent i's value for bundle S_i.

[**Correctness**] Correctness, $\mathit{corr}(\mathbf{S})$, is measured as the average number of times that an auction finds the optimal allocation. Correctness can provide a more sensitive measure of performance than efficiency.

[**Revenue**] Revenue, $\mathit{rev}(\mathbf{S})$, is measured as auctioneer's final revenue, as a ratio of the value of the optimal solution: $\mathit{rev}(\mathbf{S}) = \sum_i p_i(S_i)/V^*$ where $p_{\text{bid,i}}(S_i)$ is the final payment

made by agent i for bundle S_i.

A simple metric is used to assess information revelation in the auction.

[Information Revelation] Information revelation, $inf(i)$, for agent i is measured as the sum of the final price bid by the agent for all bundles in its valuation function, as a fraction of the sum of the true value of each bundle:

$$inf(i) = \frac{\sum_{S \in \text{bid}_i} p^*_{\text{bid,i}}(S)}{\sum_{S \in \text{val\_fun}_i} v_i(S)}$$

where $p^*_{\text{bid,i}}(S)$ is the *maximum* bid from agent i for bundle S during the auction; bid_i is the set of bundles that receive bids from agent i; and val_fun_i is the set of bundles with positive value in agent i's valuation function.

The overall information revelation, inf, is computed as the average information revelation over all agents. Asymptotically, if the auction terminates after the first round, as is would in the example in Figure 3.1 in Chapter 3, $inf = 0\%$, while if the auction terminates only after every agent has revealed its complete value for all bundles in its valuation function, $inf = 100\%$. The information revelation in the GVA is clearly 100%.

In addition to reducing information revelation, we would also like an agent to be able to follow a myopic best-response strategy in an iterative auction without computing its exact value for all bundles. Chapter 8 considers the complexity of an agent's best-response bidding problem. In simple terms, best-response is possible without exact values on all bundles because an agent must only determine the bundle with a utility (value - price) that *dominates* the utility of the other bundles at the current prices. This can be determined with appropriate upper- and lower- bounds on the value of each bundle. In this chapter I take information revelation as a proxy for agent valuation. This is reasonable in the limit because complete information revelation certainly requires that an agent computes exact values for all bundles.

[Communication Cost] The communication cost is a measure of the number and size of messages sent between agents and the auctioneer during an auction. A bundle is defined with $|\mathcal{G}|$ bits, an item with $\log_2 |\mathcal{G}|$ bits, and a value with $\log_2 V_{\max}$, where $V_{\max}$ is the largest possible value (for integer values). In *i*Bundle both price increases and best-response bids require that only the bundle is specified, which costs $|\mathcal{G}|$ bits. This is because the bid price on a bundle is exactly the ask price in myopic best-response, while the amount of a price increase is always ϵ. As a fraction of the communication cost in the GVA, we

measure the total cost to an agent in *i*Bundle as:

$$comm_i = \frac{|\mathcal{G}|(n_{\text{bid}} + n_{\text{price}})}{|V_i||\mathcal{G}| \log_2 V_{\max}}$$

where there are n_{bid} bids, n_{price} increases, and $|V_i|$ bundles with non-zero value in the agent's valuation function. In some implementations the price messages may be broadcast, in which case this communication cost is atomized across all agents.

5.3.2 Problem Sets

*i*Bundle is tested on several problem sets from the literature. The problem set characteristics are summarized in Table 5.5. The CalTech problem set [LPR97] is designed to represent a hard spatial fitting problem, and has been used to test the AUSM and RAD bundle auctions [DKLP98]. Problem sets PS1 and PS2 are resource allocation problems that have been used to test the performance of a sequential auction mechanism (SEQ) with adaptive agent bidding strategies [BGS99].

Problem sets 4-8 are designed to represent different levels of subadditivity and superadditivity over items. I refer to these problem sets as *k-comp(g)*. Agents have subadditive values for combinations of items when $k < 1$, and superadditive values when $k > 1$. The parameter g indicates how many items are covered by bundles with positive value in each agent's valuation function.

Problem sets 9-16 are generated from bid distributions used to test a new winner determination algorithm for bundle auctions [San99]. Agent valuation functions are generated by partitioning the bid distributions across the agents. In problem sets 9–12 agents have exclusive-or values across bundles, i.e. $v_i(S \cup T) = \max(v_i(S), v_i(T))$, while in problem sets 13–16 agents have additive-or values, i.e. $v_i(S \cup T) = v_i(S) + v_i(T)$ for disjoint bundles $S \cap T = \emptyset$.

Larger problem sets than those in Table 5.5 are constructed from Sandholm's Decay, Random, Uniform, and Weighted-Random distributions, increasing the number of agents and/or the number of items. In our main experiments the number of items, $|\mathcal{G}| = 50$, and I scale the problems by increasing the number of agents, $|\mathcal{I}|$, from 5 to 40, with values for 10 bundles per agent. Sandholm's α parameter in the Decay distribution is set to $\alpha = 0.85$, and bundles of size 10 are computed in the Uniform distribution.

Table 5.5 states the number of items $|\mathcal{G}|$ in each problem set, the number of agents $|I|$, the average number of bundles with positive value for each agent, and whether the agents

Problem #	Name	$\|\mathcal{G}\|$	$\|I\|$	Number bundles per agent	(X)or / (O)r	Num agents in sol (%)	Naive *eff* (%)	Naive *corr* (%)	Num trials
1	*CalTech*	6	5	5	X	40.0	63.2	2	50
2	*PS1*	12	4	3.97	X	89.0	82.1	20	50
3	*PS2*	12	5	4.07	X	58.4	79.3	20	50
4	*0-comp(4)*	5	5	15	X	85.6	61.2	0	50
5	*0.5-comp(4)*	5	5	15	X	80.8	63.2	0	50
6	*1-comp(4)*	5	5	15	X	71.2	63.0	0	50
7	*2-comp(4)*	5	5	15	X	49.2	65.3	4	50
8	*4-comp(4)*	5	5	15	X	43.6	63.5	6	50
9	*random*	10	5	10	X	84.8	64.9	8	25
10	*w-random*	10	5	10	X	38.4	82.8	20	25
11	*uniform*	20	5	10	X	60.0	73.0	8	25
12	*decay*	20	5	10	X	96.0	80.2	12	25
13	*random-or*	10	5	10	O	74.4	55.3	0	25
14	*w-random-or*	10	5	10	O	39.2	82.4	20	25
15	*uniform-or*	20	5	10	O	48.8	69.6	4	25
16	*decay-or*	20	5	10	O	92.8	72.5	0	25

Table 5.5: Problem characteristics.

have OR or XOR values over bundles. Table 5.5 also records the average percentage of agents in the optimal allocation. All other things being equal, we would expect a greater proportion of agents to receive bundles as the number of items increases, the number of agents decreases, and the level of superadditivity decreases. For example, the number of agents in the optimal solution falls as k increases in the *k-comp(g)* problem sets. I also compute the performance of a naive central algorithm `naive` for each problem set, to provide a baseline for the performance of the auction-based solutions. The naive algorithm repeatedly selects an agent at random (without replacement) and tries to allocate bundles to the agent until it is happy, choosing bundles in order of decreasing value.

5.3.3 Comparison Auction Mechanisms

I compared the performance of *i*Bundle with reported results for other auctions. AUSM and RAD are iterative auctions that allow agents to bid for bundles [LPR97, DKLP98]. SEQ [BGS99] is a sequential auction, in which each item is sold in a sealed-bid auction in sequence, and agents must learn to anticipate the future prices of items that they need to complete a bundle when bidding for early items.

I also implemented a simple simultaneous ascending price auction, with and without bid withdrawal (SAA-w and SAA). In SAA-w agents can withdraw a bid in any round.

When an agent withdraws a bid (j, p) for item $j \in \mathcal{G}$ at price p the ask price for the item in the next round is set to p. If the item remains unsold the agent must pay its bid price, but otherwise there is no penalty. This approximates the rule used in the FCC spectrum auction [Plo97]. The best-response bidding strategy in SAA is to bid for items that maximize utility, assuming they will win every bid. Without budget constraints agents write-off incomplete bundles with this strategy, in a phenomena described by Bykowsky *et al.* [BCL00] as *mutually destructive bidding*.

The best-response strategy in SAA-w is similar, except that agents assume that they can decommit for free. Once an agent has withdrawn a bid, the penalty represents a sunk cost. There is continued debate about the effect of bid withdrawal on auction performance [BCL00, Por99].

5.3.4 Experimental Platform

*i*Bundle is coded in C++, with a branch-and-bound depth-first search used to solve the auctioneer's winner determination problem in each round. Modules to generate random problem sets, and simulate agent bidding strategies were also coded in C++. I have also experimented with the GAMS/CPLEX platform as a method to solve winner-determination in each round of the auction.

A variation on Sandholm's depth-first branch-and-bound search algorithm [San99] solves winner-determination in each round, and computes the allocation and prices in the GVA. A new heuristic is introduced to make search more efficient for XOR bids. The heuristic computes an overestimate of the possible value of a partial allocation based on allocating at most *one* bundle to each remaining agent without a bundle.

5.3.5 Normalized Bid Increment

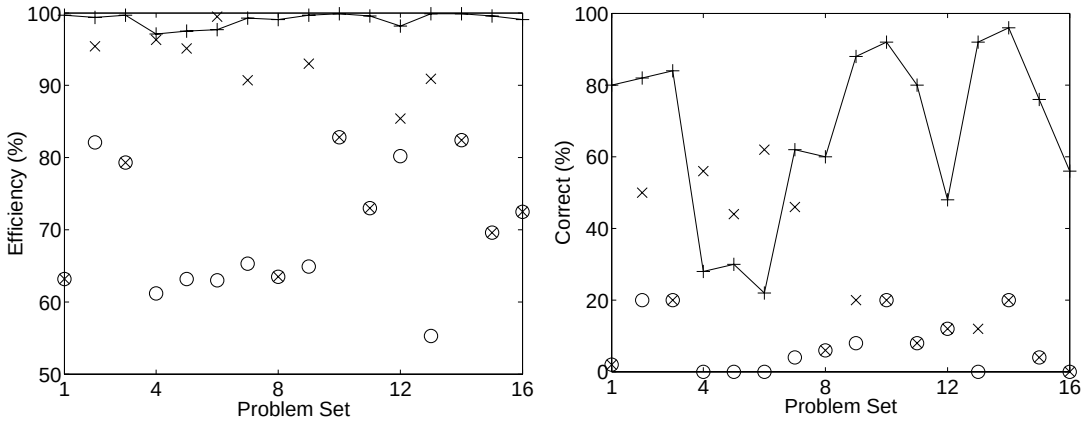
In some tests it is useful to normalize the minimal bid increment across problem distributions, to give some consistency in comparisons of allocative efficiency, etc. The minimal bid increment ϵ is adjusted to normalize for the number of bundles in an average solution and the average value of an optimal solution; i.e. an increment of $x\%$ represents an actual bid increment $\epsilon = xV^*/(100W^*)$ where W^* is the average number of bundles in the optimal allocation and V^* is the average value of the optimal allocation.

Problem #	Problem Name	Performance measure	SEQ	RAD	AUSM	*i*Bundle(2) ϵ (%) 20	*i*Bundle(2) ϵ (%) 5
1	*CalTech*	*eff*		90.4	94	96.4	99.7
		corr		80		36	80
		rev		79	71	70.6	77.7
2	*PS1*	*eff*	87			92.4	99.4
3	*PS2*	*eff*	80			92.8	99.7

Table 5.6: Performance comparison with SEQ, RAD and AUSM on problems 1, 2 and 3. Bid increment ϵ (%); Efficiency *eff* (%); Correctness *corr* (%); Revenue *rev* (%).

5.4 Results I: Efficiency and Information Revelation

Table 5.6 compares the performance of *i*Bundle(2) with reported results for AUSM and RAD [DKLP98, LPR97] on problem set 1 in Table 5.5. The experiments reported in DeMartini *et al.* [DKLP98] are with human participants, and it is possible that software agents could perform better (or worse). This aside, *i*Bundle(2) achieves a higher efficiency than RAD and AUSM, and is competitive in revenue. I also compared the performance of *i*Bundle with SEQ [BGS99] on problem sets 2 and 3. *i*Bundle(2) generates almost perfect allocations, significantly outperforming SEQ (results on *corr* and *rev* are not available for SEQ). The empirical results reported for SEQ are with agents that follow sophisticated bidding strategies, learned over many repeated trials of the same problem instance. In comparison, *i*Bundle agents follow simple best-response bidding strategies.



(a) (b)

Figure 5.2: Performance of SAA-w 'x', *i*Bundle(2) '+', and a naive central resource allocation algorithm 'o'. Bid increment $\epsilon = 5\%$. (a) Efficiency. (b) Correctness.

Problem		Perf	*i*Bundle(2)				*i*Bundle(d)				*i*Bundle(3)			
#	Name	measure												
1	*CalTech*	ϵ	5	2	1	0.5	5	2	1	0.5	5	2	1	0.5
		eff	100	100	100	100	100	100	100	100	100	100	100	100
		corr	94	96	98	**100**	94	98	**100**	**100**	94	98	**100**	**100**
		inf	87.6	89.8	90.4	91	87.6	89.8	90.4	91	87.6	89.8	90.4	91
2	*PS1*	ϵ	5	2	1	0.5	5	2	1	0.5	5	2	1	0.5
		eff	98.3	99.7	99.9	99.8	98.3	99.8	100	100	98.3	99.8	100	100
		corr	65.8	91.1	98.7	97.5	65.8	92.4	**100**	**100**	65.8	92.4	**100**	**100**
		inf	49.2	42.4	40.8	39.9	49.2	43.5	41.7	40.9	49.2	43.6	41.9	41.1

Table 5.7: Achieving optimal solutions with *iBundle*, problems 1 and 2. Bid increment ϵ; Efficiency *eff* (%); Correctness *corr* (%); Information revelation *inf* (%).

Figure 5.2 plots the efficiency (a) and correctness (b) of *i*Bundle(2) and SAA-w (with $\epsilon = 5\%$) for each problem set, together with the `naive` performance as a baseline. The SAA auction fails in many problem sets (1, 3, 8, 10–16), in the sense that agents *lose* utility through participation when the best-response bidding strategy leaves them with incomplete bundles. The efficiency of SAA is somewhat misleading in these problems because it indicates a significant problem with myopic best-response as an agent bidding strategy. SAA-w allows bid withdrawal, and mitigates the exposure problem in problem sets 12 and 13.

It is useful to consider average performance across all problem sets. For the sake of analysis I substitute the efficiency and correctness of `naive`, and the revenue from the Generalized Vickrey Auction (GVA), in problem sets where SAA and SAA-w fail. The naive central algorithm provides a useful lower bound on efficiency and correctness, but revenue is undefined. The GVA provides a lower bound on revenue, for agents that follow rational bidding strategies in an auction that generates efficient solutions.

*i*Bundle(2) (with $\epsilon = 5\%$) achieves an efficiency of 99%, compared to 83.3% efficiency for SAA-w, 81.6% for SAA, and 70.1% for `naive`. SAA-w performs well in problem set 2 (where there is little competition for resources), and sets 4, 5 and 6, where agents have subadditive, linear-additive or slightly superadditive values on bundles. Average correctness is 67.2% for *i*Bundle(2), 23.9% for SAA-w, and 7.8% for `naive`. Finally, *i*Bundle(2) generates 75.6% revenue, compared to 70.8% for SAA-w and 62.9% for the GVA.

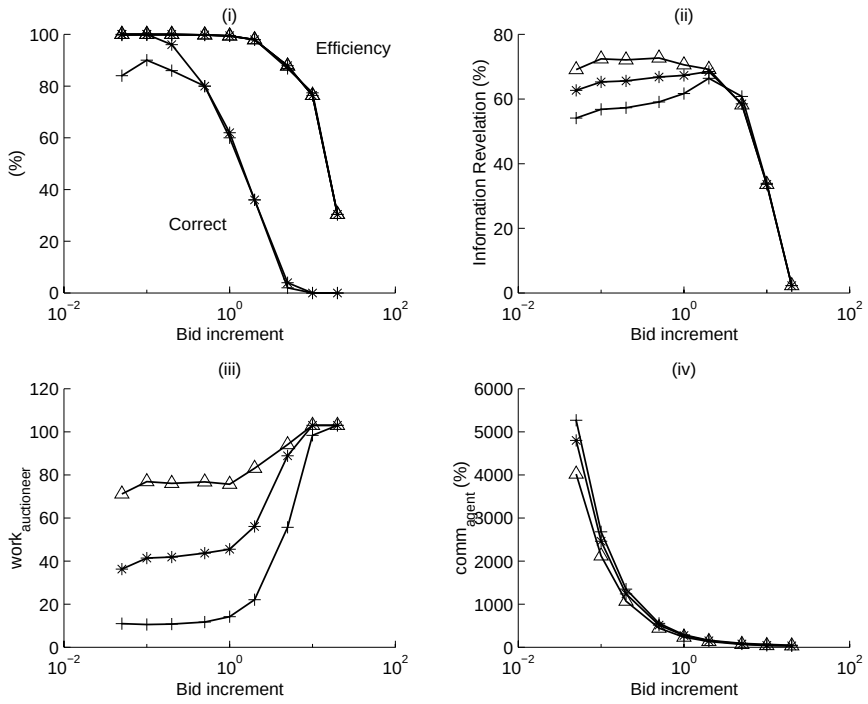


Figure 5.3: Performance of *iBundle* as the bid increment ϵ decreases. '+' iBundle(2); '$*$' iBundle(d); 'Δ' iBundle(3). Problem *0.5-comp(3)*.

5.4.1 Effect of Price Discrimination

Figure 5.3 (i) compares the efficiency and correctness across each auction variation for problem set 0.5-comp(3), which is a problem with 5 agents, 5 items, and 7 bundles/agent. Although price discrimination is required for 100% correctness in this problem, the efficiency improvement is negligible. The increase in efficiency with *i*Bundle(d) and *i*Bundle(3), compared to *i*Bundle(2) is marginal. With $\epsilon = 5\%$, *i*Bundle(3) achieves 99.1% efficiency, compared to 99% efficiency for *i*Bundle(2). Table 5.7 presents similar results for Problems 1 and 2.

Price discrimination only makes a difference for very small bid increments, when the communication cost begins to increase rapidly. For bid increment $\epsilon > 0.5\%$ the performance is almost identical. The relationship between average number of rounds to termination and bid increment is approximately linear, with $\epsilon = 5\%, 0.5\%, 0.05\%$ corresponding to 6, 49, and 480 rounds respectively. An auctioneer may choose not to operate below 0.5% because of high communication costs, computation costs, and indirect costs due to elapsed time.

Figure 5.3 (ii) plots the information revelation; (iii) the auctioneer's computation cost; and (iv) the agent's communication cost. The computation cost in this experiment is assumed to scale exponentially with the worst-case size of the winner-determination problem solved in *i*Bundle, and with the size of the winner-determination problem with all agents in the GVA. Size is measured as the product of the number of bids received (or bundles with values in the GVA) and the number of items. Finally, the cost is normalized on a log-scale, with the cost of the GVA equaling 100%. We might expect the worst-case runtime in *i*Bundle to be dominated by the time on the largest problem in any round because winner-determination is NP-hard. Similarly, the cost of solving WD for all agents should dominate in the GVA.

The results show that *i*Bundle can tradeoff performance for communication, computation, and information revelation costs. *i*Bundle(2) achieves allocations that average 91.7% efficiency across all problem sets and terminates after 5.7 rounds with bid increment $\epsilon = 20\%$, down from 99% efficiency after 18 rounds with bid increment $\epsilon = 5\%$. Price discrimination only helps for bid increments less than around $\epsilon = 0.5$, at which point the communication cost begins to increase rapidly. It is likely that an auctioneer would choose not to operate in this region anyway. The performance of all auction variations is approximately identical in the region with reasonable communication cost. As the bid increment decreases we can achieve a continuous tradeoff between performance (efficiency and correctness), information revelation, and communication cost. Notice also that the dynamic and third-order *iBundle* auctions require more information revelation and auctioneer computation than the second-order *iBundle* auction in this problem domain, but have less communication cost.

5.4.2 Information Revelation

*i*Bundle(2) requires an average of 57.5% information revelation at $\epsilon = 20\%$ (when the allocations are 91.7% efficient), and an average of 71% information revelation at $\epsilon = 5\%$ (when the allocations are 99% efficient). The (sealed-bid) GVA requires 100% information revelation from agents to achieve 100% efficiency. I would expect information revelation to be smaller in real-world problems. The agents in the problem sets have *sparse* valuation functions, which limits the size of the worst-case information revelation, i.e. the denominator in the information-revelation metric. In addition, information revelation would be smaller in

Problem #	Name	$\|\mathcal{G}\|$	$\|I\|$	Number bundles per agent	(X)or / (O)r	Num agents in sol (%)	Naive *eff* (%)	Naive *corr* (%)	Num trials
17	*decay(2)*	20	2	25	X	100	93.0	64	25
18	*decay(10)*	20	10	5	X	77.2	75.5	4	25
19	*decay(25)*	20	25	2	X	37.4	61.2	0	25
20	*decay(50)*	20	50	1	X	20.6	62.8	0	25

Table 5.8: Problems for the easy-hard scalable performance test.

"easier" problems, for example with significant over-demand or significant under-demand. In the former only a few agents need to drive the price adjustment, while in the latter we would expect *i*Bundle to terminate quickly because it is likely that an early provisional allocation would include bids from all agents.

One of the main claims for iterative auctions vs. sealed bid auctions is that they provide major computational advantages in easy problem instances, terminating quickly with little information revelation and computation. To test this claim I studied the *Decay* problem set as the number of agents increased from 2 to 50. Table 5.8 summarizes the statistics for the problems. The problem is relatively easy with a few agents, but as the number of agents increases the level of competition increases and it is more difficult to compute the optimal solution. For example, all agents are in the optimal allocation with a few agents, and the efficiency and correctness of the *naive* algorithm is high.

Figure 5.4 illustrates the performance of *i*Bundle as the "difficulty" of the combinatorial allocation problem is increased. In each trial I select a bid increment to make *i*Bundle compute a solution with the same quality, i.e. with a similar efficiency and correctness, see Figure 5.4 (i).

As the instances get more difficult the auctioneer's computational cost steadily increases, see Figure 5.4 (iii). As described above, this is measured as the *size* of the largest winner-determination problem the auctioneer must solve, as a fraction of the size of the problem in the GVA (with full valuation functions from each agent). In easy problems the largest winner-determination problems in *i*Bundle are about half the size of the problems in the GVA, which could be significant for an NP-hard problem.

Communication cost in *i*Bundle(2) and *i*Bundle(d) is a linear function of the number of agents, see Figure 5.4 (iv), essentially because a cost is accounted to send price updates to all agents. This is not a very accurate measure, as simple methods that allow agents

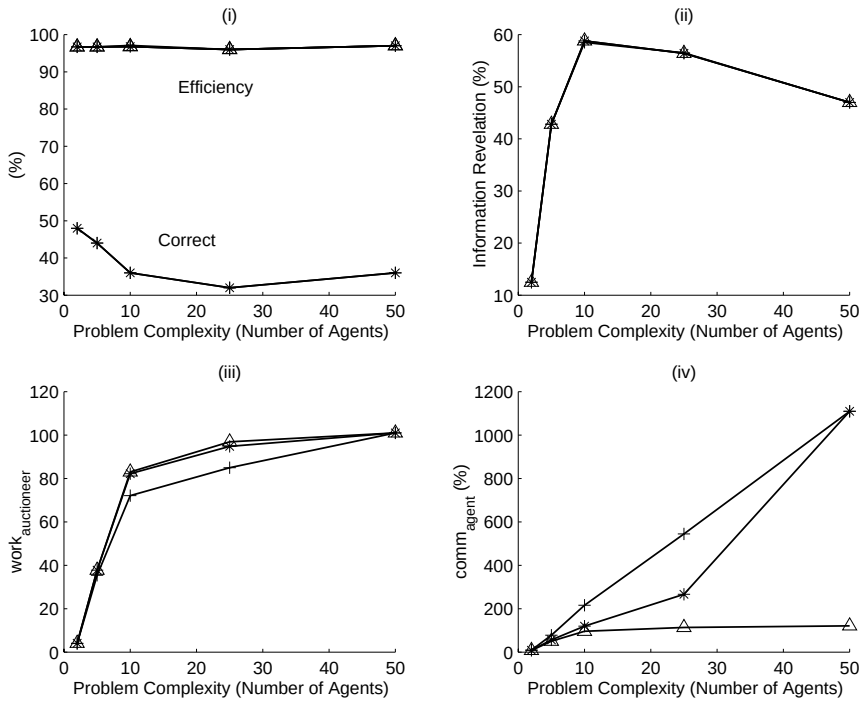


Figure 5.4: Performance of iBundle as the problem difficulty is increased. '+' iBundle(2); '$*$' iBundle(d); 'Δ' iBundle(3). Decay problem set, for 2, 10, 25, and 50 agents.

to drop out of the auction will give similar communication costs across *i*Bundle(2), (d) and (3). Essentially, the communication scaling properties are strongly sublinear in the number of agents, because many agents quickly drop out as prices get too high.

Figure 5.4 (ii) plots the effect on information revelation in *i*Bundle as the problem complexity varies. The most interesting effect can be observed between 0 and 10 agents. For less than 10 agents the problem is solved with very little information from agents. The peak information revelation occurs for problems of intermediate size, when every agent in the system must report quite accurate and complete information. As the number of agents increase the average information revelation falls because information from some agents becomes irrelevant.

5.5 Results II: Winner Determination and Communication Cost

The auctioneer must solve one WD problem in each round, and a naive worst-case analysis gives $O(BV_{\max}/\epsilon)$ rounds to converge, for a total of B bundles with positive value over all agents, maximum value $V_{\max}$ for any bundle, minimal bid increment ϵ, and myopic best-response. In the worst-case the price of a single bundle must increase by at least ϵ in each round the auction remains open, and prices are bounded by the maximum value over all agents. The number of rounds to termination is inversely proportional to the minimal bid increment.

The auctioneer announces only price *increases* in each round, and does not maintain explicit prices for all possible bundles. Instead, bid prices are verified dynamically in each round, to check that bids are at least as large as the ask price of all contained bundles. The total work in checking each bid is *linear* in the number P of bundles that have explicit ask prices, with a naive linked-list data structure. Similarly, price monotonicity can be maintained in linear time in P for each new price increase. In addition, $P \leq B$, with agents that have values for B bundles, because only bundles that receive bids can receive explicit ask prices.

The winner-determination (WD) problem that the auctioneer solves in each round of *i*Bundle is $\mathcal{NP}$-hard, itself a small instance of the CAP problem. However, each problem is smaller than the problems in the GVA, because the winner-determination problem is revenue-maximization given agents' best-response bids, while in the GVA the problem is to compute the efficient allocation directly from valuation functions. However, the problem instances might also be hard instances because all agents bid at similar prices for bundles (Andersson *et al.* [ATY00]), which can restrict the ability of search-based methods to prune the search space.

The following sections describe a number of interesting methods to speed-up the winner-determination problem in *i*Bundle. First, the provisional allocation from the previous round provides a good initial solution to the WD problem, because agents must re-bid bundles received in the previous round. This allows pruning of the search for a revenue-maximizing allocation. An additional saving in computation time is achieved by limiting search to an allocation at least ϵ better than the value of the allocation in the previous

round. Approximations can be introduced to speed-up winner-determination, both through increasing the minimal bid increment and decreasing the number of winner-determination problems to solve (one in each round), and with approximate winner-determination algorithms. Another approach is to *cache* provisional allocations from previous rounds, and use cached solutions to compute good initial solutions in the current round of the auction. One interesting heuristic suggested by an analysis of the hit statistics of the cache is the "flip-flop" heuristic, which adopts a cache hit as the new provisional allocation without additional search.

5.5.1 Minimal Bid Increment Approximations

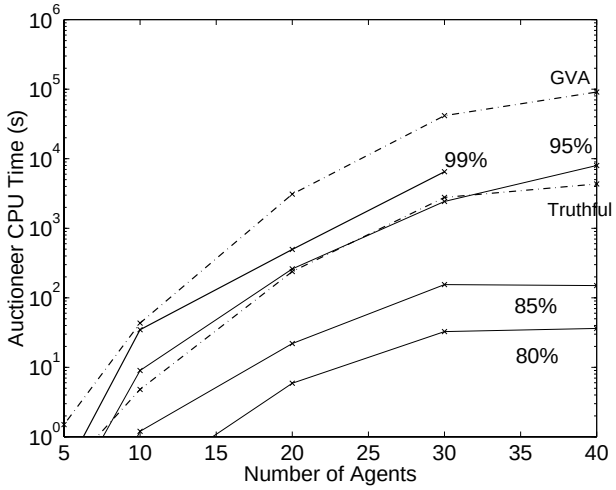


Figure 5.5: Total computation time in *i*Bundle(*2*), the GVA, and a sealed-bid auction with truthful agents, in problem set Decay. The performance of *i*Bundle is plotted with different bid increments ϵ, selected to give allocative efficiency of 80%, 85%, 95% and 99%.

One method to introduce approximations is via the minimal bid increment. Figure 5.5 plots the total auctioneer winner-determination and price-update time[2] in *i*Bundle in the Decay problem set. Performance is measured for different bid increments, with the bid increment selected to give allocative efficiency of 80%, 85%, 95% and 99% ($\pm$1%). Figure 5.5 also plots performance for the GVA, and for a sealed-bid auction in which agents are assumed to bid truthfully. The GVA proved intractable for 30 and 40 agents. In those problems the run time is estimated as the time to compute the optimal solution in a single

[2]Time is measured as user time in seconds on a 450 MHz Pentium Pro with 1024 MRAM, with *i*Bundle coded in C++.

WD problem multiplied by the number of agents in the optimal allocation. Results are averaged over 10 trials. First, note that the curves are sublinear on the logarithmic value axis as the number of agents increases, indicating polynomial computation time in the number of agents.

The performance improvement of *i*Bundle over GVA is striking, achieving at least one order of magnitude improvement with 99% allocative efficiency and three orders of magnitude with 85% allocative efficiency. For up to 95% efficiency I essentially get the myopic truth-revelation properties of *i*Bundle for free, because *i*Bundle's run-time is approximately the same as for the sealed-bid auction with truthful agents.

Problem		GVA	*i*Bundle			Approx-
			$\simeq 90\%$	$\simeq 95\%$	$\simeq 99\%$	Bundle
Decay	Eff (%)	100	91.5	94.9	98.3	85.1
67.3%[d]	WD-time[a] (s)	41700	831	2400	5650	0
13.4[e]	Pr-time[b] (s)	–	26	34.5	44	39.2
	Comm[c] (kBit)	18.8	221	306	394	377
WR	Eff (%)	100	90.7	94.9	99.2	79.4
71.5%	WD-time (s)	3	0.6	1.7	6	0
1	Pr-time (s)	–	5.4	11.5	40.9	12.2
	Comm (kBit)	18.1	20.5	52.1	144	53.1
Rand	Eff (%)	100	89.3	97	99	95.8
37.8%	WD-time (s)	68	4.4	7.4	11	0
11.2	Pr-time (s)	–	6.5	9.7	12.1	12.9
	Comm (kBit)	18.7	49.5	66.4	82.6	85.6
Unif	Eff (%)	100	–	95.6	99.1	76.2
58%	WD-time (s)	25	–	6.6	18.7	0
3	Pr-time (s)	–	–	14.7	42.0	46
	Comm (kBit)	18.2	–	56.5	120	124

Table 5.9: Performance in the Decay, WR, random, and uniform problems. [a] Auctioneer WD time. [b] Price-update time. [c] Communication cost. [d] Alloc. eff. of a sealed-bid auction with a greedy WD algorithm and truthful agents. [e] Average number of agents in the optimal solution.

Table 5.9 compares *i*Bundle with the GVA for all Sandholm's problems, for problems with 30 agents. These WR and Uniform problem instances are quite easy because the optimal allocation sells large bundles to a few agents, which allows considerable pruning during search. In comparison, The Random and, in particular, Decay problems tend to be harder because the optimal allocation requires coordination across a number of agents, see also Sandholm [San99] and Andersson *et al.* [ATY00]. In all problems *i*Bundle has less WD time at 95% allocative efficiency than the GVA.

Note that the price-update step is relatively expensive in the otherwise easy weighted-random (WR) problem, because bid prices for large bundles must be checked for price consistency against the price of all included bundles.

5.5.2 Approximate Winner-Determination

Another method to introduce approximation is via a greedy winner-determination algorithm in each round of the auction.

The auctioneer can maintain the same incentives for myopic agents to follow the same bidding strategy for any approximation algorithm with the *bid-monotonicity* property:

DEFINITION 5.2 [bid-monotonicity] An algorithm for winner-determination satisfies bid-monotonicity if whenever an agent i is allocated a bundle with bids $\mathcal{B}_i$, it is also allocated a bundle with bids $\mathcal{B}_i \cup B$ that include a bid for an additional bundle B.

It is straightforward to prove that *optimal* winner-determination algorithms are bid-monotonic. One simple algorithm with the bid-monotonicity property is due to Lehmann *et al.* [LOS99], which allocates bundles in order of decreasing value-per-item.

Table 5.9 presents the efficiency in *i*Bundle with this approximation algorithm for winner-determination. Notice that *i*Bundle computes a fast solution to all problems with the approximate winner-determination algorithm, even in the hard Decay problem set. The average allocative-efficiency of *i*Bundle with the approximate winner-determination algorithm in Decay is a quite respectable 85.1% (notice that a greedy centralized solution with truthful agents achieves only 67.3%), and with more than a 1000-fold reduction in winner-determination time. I believe that other, slightly less greedy, approximate algorithms will give even further performance improvements.

5.5.3 Methods to Speed-up Sequential Winner-Determination

The sequential nature of winner-determination in *i*Bundle suggests that cache-based methods can speed-up winner-determination. I experimented with a cache of all provisional allocations computed in previous rounds. The auctioneer first checks the cache when new bids are submitted. This provides an initial solution for winner-determination and is used to prune search. A simple linear program is used to select the best allocation from the cache.

Problem		WD Time				% Cache Correct			
		0	1	T	$T!$				
Decay	50/15/150[a]	415	371	355	291	0	28	47	59
WR	50/50/1000	253	243	231	163	0	11	57	57
Rand	50/30/600	1823	1616	1491	864*	0	6	30	78
Unif	50/40/800	343	337	336	110	0	14	29	49

Table 5.10: Winner-determination time with caches of size 0, 1 (last round), and T (all previous rounds). In cache $T!$ revenue maximizing cached solutions from previous rounds are assumed optimal. *Eff* $> 99\%$ in all problems except *, where *Eff* $= 96.8\%$. [a] $|\mathcal{G}|$ / $|I|$ / # bundle values.

Table 5.10 compares the WD time in each problem with and without caching of previous allocations. Although the overhead time required to maintain and test the cache is small, the effect of even a complete cache on all previous rounds is not very dramatic. The problem is that the hard part of solving a winner-determination problem, and an instance of the CAP, is not *finding* the optimal solution, but *verifying* that the solution is optimal. A straightforward use of a cache might help to find the optimal solution, but does not speed-up the search required to verify that solution, For example, although an extended cache in the Decay problem provides the correct allocation in 47% of problems, the speed-up is limited to around 14%.

In an attempt to leverage the correct solutions from the cache, I tested the performance of *i*Bundle under an additional assumption that if a cached solution from before round $t \Leftrightarrow 1$ generates more revenue than the solution from round $t \Leftrightarrow 1$, this is adopted as the new provisional allocation without further computation. The rule is designed to capture "flip-flop" competition between a number of good allocations during an auction.

Labeled $T!$, the rule proves useful in Decay, WR and Uniform, reducing computation by 30%, 36% and 68% from the time with no cache for a negligible drop in allocative efficiency. However, one must be careful: although we also see a speed-up in Random, the allocative efficiency falls from 99% to 96.8%. Further analysis shows that cached solutions prove optimal in 54%, 97% and 49% of rounds in Decay, WR and Uniform, but only optimal 34% of rounds in Random.

Further optimizations should be possible, for example using cached solutions once a large enough cache is constructed, and solving WD when an auction is about to terminate with cached solutions. Another useful approach is ϵ-scaling, that adjusts the bid increment during an auction [Ber90].

5.5.4 Communication cost

Communication cost is computed using the metric specified in Section 5.3.1, with 10 bits to specify a value in the GVA. Cost is measured as $|\mathcal{G}|n_{\text{bids}}$ for each agent plus $|\mathcal{G}|n_{\text{prices}}$, given n_{bids} bids and n_{prices} prices. I assumed that prices are broadcast to agents, and assess the cost of price information independently of the number of agents in an auction. Only the cost of bids is counted in *i*Bundle(3), with non-anonymous prices, because the price changes are implicit in the new provisional allocation. An agent's new prices are ϵ above its bid price whenever its bids are unsuccessful.

There is a communication cost penalty in using *i*Bundle compared with the GVA in these problems (Table 5.9) because of repeated bids across a number of rounds. This would change in problems with agents that have values for many bundles because all values must be reported in the GVA, or in easier problems because *i*Bundle will terminate quickly with less bids.

5.6 Special Cases for Expressive Bid Languages

It is possible to derive fast special cases of *i*Bundle for restrictions on agent bidding languages, which solve the combinatorial allocation problem for assumptions about agent preferences. The advantage of identifying special cases is that the winner-determination problem is tractable for particular structures on bundles and price functions over bundles (see Section 4.5).

5.6.1 Unit-Demand Preferences (Assignment Problem)

In problems where agents have single-unit demands (the standard assignment problem) we can restrict *i*Bundle in two ways (leading to two different auction mechanisms):

- Restrict agents to placing OR bids on items. In this case *i*Bundle(2) reduces to a simultaneous ascending-price auction, that was shown to be optimal for the assignment problem by Crawford & Knoer [CK81].

- Restrict agents to placing XOR bids on items. In this case the winner-determination problem in *i*Bundle(2) can be formulated as a max-flow problem, and solved efficiently.The auction only generates prices on items, and is a variant on the auction

proposed by Demange *et al.* [DGS86].

5.6.2 Linear-Additive Preferences

In problems where agents have linear-additive demands on items we can restrict agents to OR bids on items. The auction reduces to a simultaneous ascending-price auction, equivalent to Crawford & Knoer's [CK81] auction for the problem.

5.6.3 Gross Substitutes Preferences

Gross substitutes (GS) preferences implies that an agent that demands bundle S in round t will continue to demand items in S that do not increase in price as the price of other items increases [KC82]. Linear-additive and unit-demand preferences are special-cases of gross substitutes. Kelso & Crawford prove GS are a sufficient condition for the existence of linear competitive equilibrium prices. Again, we can restrict agents to OR bids on items:

- Agents bid for all the items in the single bundle that maximizes their utility in each round. Gross substitutes implies that an agent will continue to demand any subset of the items as the price of other items increases, and be happy to continue to bid for items it receives in a partial allocation. In this case *i*Bundle(2) reduces to a simultaneous ascending-price auction, that was shown to be optimal for GS by Kelso & Crawford [KC82].

5.6.4 Multiple Homogeneous Goods

The multiple homogenous good allocation problem is to allocate $|\mathcal{G}|$ identical items across agents, to maximize the total value over all agents. Each agent has a valuation function $v_i(m) \geq 0$ defined for (integer) quantities $m \geq 0$ of the item.

A suitable bidding language allows an agent to submit exclusive-or bids of type (m, p), which states that the agent will pay at most p for a bundle of at least m items. Essentially, each bid represents a set of exclusive-or bids over all bundles of items in $\mathcal{G}$ of size m. The winner-determination problem is a multi-unit allocation problem, where each agent must receive exactly the number of items in its bid. Finally, prices are increased on unsuccessful bids, with prices maintained for each possible size of bundle (with non-anonymous prices of the same kind also introduced dynamically). Again, we do not want to maintain an explicit price for each distinct set of bundles of the same size.

An unsuccessful bid at price p on bundles of size $m > (|\mathcal{G}|/2)$ indicates that the minimal bid price on *all* bundles of size m must be at least $p+\epsilon$ in future rounds. It is not necessary to announce, or maintain, explicit prices on all possible bundles of size m, but rather just announce a price on bundles of size m. Notice that $m > (|\mathcal{G}|/2)$ implies the *safety* condition holds over the underlying exclusive-or set of bundles.

An unsuccessful bid at price p on bundles of size $m \leq (|\mathcal{G}|/2)$ must be handled a little more carefully because it remains possible for the same bid from two different agents at that price to be successful. We must introduce separate prices for this agent, and increase its ask price on bundles of size m by ϵ. Notice that $m \leq (|\mathcal{G}|/2)$ implies the *safety* condition does *not* hold over the underlying exclusive-or set of bundles.

Effectively, we might expect bids in this auction design to start-off anonymous while agents bid for large bundles of items at low prices, and then become more non-anonymous as agents begin to bid for smaller bundles as the prices increase on large bundles. Eventually, it is quite possible that the auction will terminate with different prices to different agents for bundles of the same size. The exact details of this auction, and an efficient implementation, are left for future work.

5.7 Earlier Iterative Combinatorial Auctions

Rassenti *et al.* [RSB82] propose a sealed-bid bundle auction for the problem of allocating airport slots, where airlines value take-off and landing slots in pairs. Agents submit sealed XOR bids for bundles of take-off and landing slots. The auction computes *linear prices* that approximately clear the market, given agent bids. Finally, agents can place *bids* and *asks* for individual slots in a secondary market, to cleanup their final allocation. Although the auction design is fairly ad-hoc, empirical results with human bidders suggest that the market can achieve high efficiency with experienced bidders.

The recent FCC spectrum auction generated a lot of debate among economists about combinatorial auction mechanisms. Spectrum licenses have non-additive value in bundles because of network synergies from spatially-coherent geographical regions. The final FCC auction design was a variant on a simultaneous ascending-price auction that allowed agents limited decommitment rights, and placed participation constraints on agents to enable information exchange via prices during the auction [MM96]. The goal was to allow agents

to find a good "fit" between their demand sets and the demand sets of other bidders, and win coherent bundles of spectrum licenses.

Bykowsky *et al.* [BCL00] demonstrate the *exposure* and *existence* problem with linear prices for the combinatorial allocation problem, and study the AUSM auction [BLP89] as an example of a bundle auction to address the problem. AUSM allows agents to bid for arbitrary bundles of items, and maintains a revenue-maximizing allocation. There are no pricing rules, and agents must coordinate their own bids. Theoretical analysis is difficult because of the flexible auction rules, but see Milgrom [Mil00b]. AUSM has reasonable performance empirically, see Ledyard *et al.* [LPR97]. Bykowsky *et al.* identify the "threshold problem" for bundle auctions, where smaller bidders must coordinate bids to outbid a larger bidder. Although *i*Bundle solves this problem for agents that follow best-response bidding strategies, the effect of alternative agent bidding strategies on the threshold (or coordination) problem is unknown.

Recently, DeMartini *et al.* [DKLP98] proposed RAD, an auction that allows agents to place XOR bids on bundles but generates prices on items. Although promising empirical results have been presented, there are no theoretical results on its allocative efficiency. RAD also borrows from the FCC auction design, agents must re-submit winning bids, and there are activity rules to encourage information revelation early in the auction and encourage coordinate bidding.

Wurman's A*k*BA combinatorial auction was discussed in Section 4.7. It allows exclusive-or bids on bundles of items in each round, and maintains anonymous prices on bundles. A linear-program based price-update rule is used to adjust prices across rounds.

Appendix: Dynamic Price Discrimination

In the Appendix to this chapter I illustrate the dynamic rule for introducing non-anonymous prices in *i*Bundle on a simple set of parameterized problems, which demonstrate the importance of price-discrimination to achieve allocative-efficiency in some problems.

Consider Problem 6 in Table 5.11, in which the value of agent 2 for bundle BC, $v_2(BC)$, can take integer values between 5 and 10. The values for bundles not explicitly listed are consistent with free disposal of items, i.e. $v(S') \geq v(S)$ for all $S' \supset S$. The optimal allocation is $([1, AC], [2, B])$ for every value $v_2(BC)$ in this range. The problem is a hard

coordination problem because: agent 1 values B more highly than agent 2, but the optimal solution allocates B to agent 2; and both agents value BC more than any other bundle but receive another bundle in the optimal allocation.

	A	B	AC	BC
Agent 1	2	9	8^*	10
Agent 2	2	5^*	3	$v_2(BC)$

Table 5.11: Problem 6. Agent values, with $5 \leq v_2(BC) \leq 10$. Efficient allocation indicated *.

In this problem first- and second-order CE prices exist for $v_2(BC) \leq 7$, but only third-order CE prices exist for $v_2(BC) > 7$. For example, when $v_2(BC) = 6$ and $v_2(BC) = 7$, then $p(A) = 2, p(B) = 5$, and $p(C) = 2$ are CE prices. However, when $v_2(BC) = 8$, no first- or second-order CE prices exist, and $v(LP_1) = v(LP_2) = 13.5 > v(IP) = 13$. It is not possible to price BC high enough (so that agent 2 demands B but not BC) without making the auctioneer's revenue from selling A and BC together greater than its revenue from selling B and AC together. Third-order CE prices for $v_2(BC) = 8$ include $p_1 = (2, 9, 8, 10)$ and $p_2 = (0, 3, 1, 6)$ for bundles A, B, AC and BC, with other prices set high enough to satisfy $p_i(S') \geq p_i(S)$ for all $S' \supset S$.

Although *i*Bundle(2) solves the problem when $v_2(BC) = 6$, the auction *fails* when $v_2(BC) \geq 7$, terminating with allocation $([1, BC], [2, A])$. Auction *i*Bundle(d), with price discrimination, solves the problem. The auction switches to price discrimination, and separates the effect of agent 1's bids from ask prices to agent 2. Agent 2 continues to bid for B and the auction terminates with the optimal allocation $([1, AC], [2, B])$.

Table 5.12 (a) shows *i*Bundle(2) for Problem 6 in Table 5.11, for $v_2(BC) = 7$. The auction fails even though second-order CE prices exist. Price-updates are not *safe*, for example in rounds $t + 1$ and $t + 3$, so we might expect the auction to terminate with a suboptimal allocation. Agents 1 and 2 compete on bids for BC, and agent 1 also drives up the price on B and AC and prevents agent 2 from bidding for B because $v_1(BC) \Leftrightarrow v_1(B) < v_2(BC) \Leftrightarrow v_2(B)$. The auction terminates with the suboptimal allocation $([1, BC], [2, A])$, because agent 2 bids for item A but not B.

Table 5.12 (b) shows the auction for Problem 6 with $v_2(BC) = 6$. In this case the auction solves the problem. Agent 2 can bid B in all rounds of the auction because $v_2(BC) \Leftrightarrow v_2(B) \leq v_1(BC) \Leftrightarrow v_1(B)$, and the revenue to the auctioneer from allocation $([1, AC], [2, B])$ eventually exceeds the revenue from $([1, BC])$ or $([2, BC])$. The auction

Round	Prices				Bids					Revenue
	A	B	AC	BC	Agent 1			Agent 2		
$t-1$								(BC, 4.4)*		4.4
t	0	3.4	2.4	4.4	(B, 3.4)	(AC, 2.4)	(BC, 4.4)*	(BC, 4.4)		4.4
$t+1$	0	3.4	2.4	4.6	(B, 3.4)	(AC, 2.4)	(BC, 4.4)	(BC, 4.6)*		4.6
$t+2$	0	3.6	2.6	4.6	(B, 3.6)	(AC, 2.6)	(BC, 4.6)*	(BC, 4.6)		4.6
$t+3$	0	3.6	2.6	4.8	(B, 3.6)	(AC, 2.6)	(BC, 4.6)	(A, 0)	(BC, 4.8)*	4.8
$t+4$	0	3.8	2.8	4.8	(B, 3.8)	(AC, 2.8)	(BC, 4.8)*	(A, 0)*	(BC, 4.8)	4.8

(a) $v_2(BC) = 7$. *i*Bundle(2) fails.

Round	Prices				Bids					Revenue
	A	B	AC	BC	Agent 1			Agent 2		
$t-1$								(BC, 2.4)*		2.4
t	0	1.4	0.4	2.4	(B, 1.4)	(AC, 0.4)	(BC, 2.4)*	(B, 1.4)	(BC, 2.4)	2.4
$t+1$	0	1.6	0.4	2.6	(B, 1.6)	(AC, 0.4)	(BC, 2.4)	(B, 1.6)	(BC, 2.6)*	2.6
$t+2$	0	1.8	0.6	2.6	(B, 1.8)	(AC, 0.6)	(BC, 2.6)*	(B, 1.8)	(BC, 2.6)	2.6
$t+3$	0	2.0	0.6	2.8	(AC, 0.6)	(BC, 2.6)		(B, 2.0)	(BC, 2.8)*	2.8
$t+4$	0	2.0	0.8	2.8	(B, 2)	(AC, 0.8)*	(BC, 2.8)	(B, 2)*	(BC, 2.8)	2.8

(b) $v_2(BC) = 6$. *i*Bundle(2) works.

Table 5.12: *i*Bundle(2) on Problem 6. Bid incr. $\epsilon = 0.2$. Provisional allocations indicated *.

terminates with optimal allocation $([1, AC], [2, B])$.

Table 5.13 shows the performance of *i*Bundle(d) in Problem 6 (Table 5.11) for $v_2(BC) = 7$. I use '/' to indicate the prices for each agent. The effect of price discrimination is to *separate* the price increases caused by bids from agent 1 from the price increases to agent 2. In particular, agent 1's bids for B do not increase the price of B to agent 2, and agent 2 can continue to bid for B (unlike in *i*Bundle(2) on this problem). The auction terminates with the optimal allocation $([1, AC], [2, B])$.

It is interesting to examine the final prices in the auction, to check how well the auction minimizes CE prices. A method ADJUST is introduced in the next chapter which will adjust prices after *i*Bundle(3) terminates to minimal CE prices in all CAP problems. We will also begin to explore the connections between minimal CE prices and Vickrey payments.

In Problem 6, when $v_2(BC) = 7$, the auction terminates with ask prices are $p_1 = (0, 3, 2, 4)$ and $p_2 = (0, 2, 0, 4)$ for bundles A, B, AC and AB. Agent values are $v_1(2, 9, 8, 10)$ and $v_2 = (2, 5, 3, 7)$. These are *minimal* CE prices because any lower prices that maintain best-response changes the revenue-maximizing allocation from $([1, AC], [2, B])$ to an allocation of BC to one of the agents. In fact, the auction increases prices to agents while the

Round	Prices				Bids		Rev
	A	B	AC	BC	Agent 1	Agent 2	
$s-1$						(BC, 1.8)*	1.8
s	0	0.8	0	1.8	(B, 0) (AC, 0.8) (BC, 1.8)*	(BC, 1.8)	1.8
$s+1$	0	0.8	0	2.0	(B, 0) (AC, 0.8) (BC, 1.8)	(BC, 2.0)*	2.0
$s+2$	0 / 0	1.0 / 0.8	0.2 / 0	2.0 / 2.0	(B, 1.0) (BC, 2.0)*	(BC, 2.0)	2.0
$s+3$	0 / 0	1.0 / 0.8	0.2 / 0	2.0 / 2.2	(B, 1.0) (BC, 2.0)	(BC, 2.2)*	2.2
...	...				...	...	...
$t-1$						(BC, 3.6)*	3.6
t	0 / 0	2.6 / 1.6	1.6 / 0	3.6 / 3.6	(B, 2.6) (AC, 1.6) (BC, 3.6)*	(B, 1.6) (BC, 3.6)	3.6
$t+1$	0 / 0	2.6 / 1.8	1.6 / 0	3.6 / 3.8	(B, 2.6) (AC, 1.6) (BC, 3.6)	(B, 1.8) (BC, 3.8)*	3.8
$t+2$	0 / 0	2.8 / 1.8	1.8 / 0	3.8 / 3.8	(B, 2.8) (AC, 1.8) (BC, 3.8)*	(B, 1.8) (BC, 3.8)	3.8
$t+3$	0 / 0	2.8 / 2.0	1.8 / 0	3.8 / 4.0	(B, 2.8) (AC, 1.8) (BC, 3.8)	(B, 2.0) (BC, 4.0)*	4.0
$t+4$	0 / 0	3.0 / 2.0	2.0 / 0	4.0 / 4.0	(B, 3.0) (AC, 2.0)* (BC, 4.0)	(B, 2.0)* (AC, 0) (BC, 4.0)	4.0

Table 5.13: *i*Bundle(d) on Problem 6, with $v_2(BC) = 7$. Bid incr. $\epsilon = 0.2$. Provisional allocations indicated *.

allocation BC is revenue-maximizing (see Table 5.13).

Chapter 6

Linear Programming and Vickrey Payments

We have assumed up to this point that agents follow myopic best-response bidding strategies. This assumption proved useful, as it allowed the construction of a primal-dual based auction for the Combinatorial Allocation Problem (CAP). *i*Bundle is the first auction to compute efficient allocations in general CAP problem instances, even under an assumption of myopic agent rationality.

In this chapter we make this assumption of myopic best-response quite reasonable. I retain myopic best-response, but develop a primal-dual based method to compute Vickrey payments at the end of the auction. When successful this makes myopic best-response a sequentially rational strategy for an agent, given that every other agent also follows myopic best-response. Myopic best-response becomes a sequential Bayesian Nash equilibrium of the auction, for *all* priors over agent preferences. Connecting primal-dual theory back to the Groves mechanisms, and in particular in this case to the Generalized Vickrey auction, provides powerful incentive-compatibility and robustness-to-manipulation.

This approach of "LP + myopic best-Response + Vickrey" appears to provide a powerful constructive tool for useful iterative mechanism design. Indeed, Bikchandani *et al.* [BdVSV01] have recently demonstrated that many of the successful iterative Vickrey auctions known in the literature, such as Demange *et al.* [DGS86] for unit-demand and Ausubel [Aus97] for multi-item decreasing returns have interpretations as primal-dual algorithms for appropriate linear program formulations of the top-level allocation problem.

This chapter develops a primal-dual algorithm, VickAuction, to compute Vickrey payments. Chapter 7 then interprets the algorithm as *i*Bundle Extend&Adjust, which introduces a second phase at the end of the auction.

Earlier iterative Vickrey auctions, with the exception of Ausubel's designs [Aus97, Aus00] compute Vickrey payments through careful price adjustment (see Table 4.7). Prices are increased on the *minimal overdemanded set* of bundles in each round, in order to terminate in *minimal* competitive equilibrium prices, which are Vickrey payments in many problems.

The main innovations in my extended *i*Bundle approach are:

- Prices are adjusted *after* the auction terminates towards Vickrey payments. This relaxes the requirement of "minimal" price increases during the auction, because it is not necessary to terminate in the minimal competitive equilibrium solution.
- The adjusted prices need not be in competitive equilibrium, which allows Vickrey payments to be computed even when there is no competitive equilibrium solution that supports all Vickrey payments simultaneously.

Conceptually, it is useful to consider two distinct phases in VickAuction, and in the extended *i*Bundle auction presented in Chapter 7. The purpose of Phase I is to compute the efficient allocation from myopic best-response bids. The purpose of Phase II is to compute Vickrey payments from myopic best-response bids. The transition from Phase I to Phase II is designed to be hidden from bidders. Prices continue to increase monotonically across the two phases and the auction rules are unchanged. The provisional allocation continues to change from round-to-round in Phase II, although the final allocation implemented is that computed at the end of Phase I.

- Phase I is *i*Bundle(3), the auction variation with separate prices for each maintained explicitly from the first round. The allocation at the end of Phase I becomes the final allocation.
- Phase II continues from the prices at the end of Phase I, but injects additional competition into the system in the form of "dummy" agents simulated by the auctioneer to mimic continued bidding from real agents as they drop-out. Phase II terminates as soon as enough additional information has been collected from agents to compute Vickrey payments.

The bids from an agent in Phase II do not change the allocation or the price that it finally pays. The only effect of bids is to decrease the final price paid by other agents

in the efficient allocation. One possible weakness of this experimental auction design is the separation of concerns between Phase I and Phase II. In other auction designs the allocation decision is made concurrently with the payment decision. In Section 7.8 in the next chapter I discuss whether this decomposition of the problem across Phase I and Phase II might make collusion and other well-known vulnerabilities of Vickrey solutions more likely.

The outline of this chapter is as follows. First I present a linear program formulation of the Vickrey outcome, based on the third-order dual linear program for the CAP introduced in Section 4.4 of Chapter 4. The optimal dual solution that minimizes revenue to the auctioneer will compute Vickrey payments to all agents in some problems. In addition, I show that the *minimal price to each agent* over all optimal dual solutions computes the Vickrey payment in all problems. The linear program formulation leads to a primal-dual method, ADJUST, to compute minimal CE prices from a suitable set of CE prices, which is introduced in Section 6.2. I prove that the prices computed at the end of *i*Bundle are sufficient to compute minimal CE prices with ADJUST.

Section 6.3 introduces primal-dual method ADJUST*, which extends ADJUST, to compute *Vickrey payments* from a suitable set of CE prices. ADJUST* is useful in an auction context because it computes Vickrey payments without explicit information about agent valuation functions. I characterize necessary and sufficient conditions for ADJUST* to compute Vickrey payments.

Finally, Section 6.4 presents a primal-dual method, VICKAUCTION, which computes Vickrey payments and the efficient allocation from only best-response agent information. VICKAUCTION extends COMBAUCTION (see Section 4.6) with a second phase, PHASEII, that is designed to collect enough additional information from agents to adjust prices to Vickrey payments after termination. In the next chapter I introduce *i*Bundle Extend&Adjust, which is an extension of *i*Bundle, designed to implement VICKAUCTION as an auction. The main difficulty is making sure that an agent cannot detect the transition from *i*Bundle into the extended phase. Bids in the second phase affect only the prices paid by other agents, and an agent that knows it is in this bidding phase would choose to drop out, or perhaps even act collusively with other agents.

6.1 Minimal Competitive Equilibrium Prices

In Chapter 4 I introduced a hierarchy of linear program formulations for CAP. The third-order dual problem, DLP_3, characterizes all non-linear and non-anonymous competitive equilibrium prices.

This formulation is repeated below, with $x_i(S) = \{0, 1\}$ to denote whether bundle $S \subseteq \mathcal{G}$ is allocated to agent i, and $v_i(S) \geq 0$ denote agent i's value for bundle S. As before, $k \in K$ denotes a possible allocation, allocating a particular bundle to a particular agent and without giving any agent more than one bundle or allocating any item more than once. The variable $y(k) \in \{0, 1\}$ indicates which solution is implemented.

$$\max_{x_i(S), y(k)} \sum_S \sum_i x_i(S) v_i(S) \qquad [\text{LP}_3]$$

$$\text{s.t.} \quad \sum_S x_i(S) \leq 1, \quad \forall i \qquad (\text{LP}_3\text{-}1)$$

$$x_i(S) \leq \sum_{k \ni [i,S]} y(k), \quad \forall i, S \qquad (\text{LP}_3\text{-}2)$$

$$\sum_k y(k) \leq 1 \qquad (\text{LP}_3\text{-}3)$$

$$x_i(S), y(k) \geq 0, \quad \forall i, S, k$$

$$\min_{p(i), p_i(S), \pi} \sum_i p(i) + \pi \qquad [\text{DLP}_3]$$

$$\text{s.t.} \quad p(i) + p_i(S) \geq v_i(S), \quad \forall i, S \qquad (\text{DLP}_3\text{-}1)$$

$$\pi \Leftrightarrow \sum_{[i,S] \in k} p_i(S) \geq 0, \quad \forall k \qquad (\text{DLP}_3\text{-}2)$$

$$p(i), p_i(S), \pi \geq 0, \quad \forall i, S$$

The dual variable $p(i)$ associated with constraint $\sum_S x_i(S) \leq 1$ can sometimes be interpreted as agent i's *marginal product* [BdVSV01].

DEFINITION 6.1 [marginal product] The marginal product of agent i, $\text{MP}(i)$ is the difference in total value of the optimal allocation with and without agent i.

$$\text{MP}(i) = V^* \Leftrightarrow (V_{-i})^*$$

where V^* is the value of the optimal allocation, and $(V_{-i})^*$ is the value of the optimal allocation with agent i taken out of the problem.

This interpretation of $p(i)$ as the marginal product follows from considering the effect of reducing the right-hand side on the constraint for agent i to zero. The dual price conveys information about an the effect of an *incremental change* in the right-hand side value on the value of the optimal primal solution. In problems for which the price remains valid for a reduction from one to zero, $p(i) = \text{MP}(i)$.

This provides a tie-in with Vickrey payments, because an agent's marginal product is precisely its utility in the dominant strategy equilibrium of the GVA.

$$\begin{aligned} u_i(v_i, v_{-i}) &= v_i(S_i^*) \Leftrightarrow p_{\text{vick}}(i) \\ &= v_i(S_i^*) \Leftrightarrow ((V_{-i})^* \Leftrightarrow V_{-i}^*) \\ &= \text{MP}(i) \end{aligned}$$

where V_{-i}^* denotes the value V^* minus the value of the optimal solution with all agents in the problem to agent i, and S_i^* is the bundle allocated to agent i in the efficient allocation.

Thus, when $p(i) = \text{MP}(i)$ then $p_i(S_i^*) = p_{\text{vick}}(i)$ because dual variable $p_i(S_i^*)$ is interpreted as the price to agent i for bundle S_i^*. In general the optimal dual solution to [DLP$_3$] is not unique. The value of the dual is computed as the sum of the utility over all agents $\sum_i p(i)$ and the auctioneer's revenue π. This provides some flexibility in prices $p_i(S)$ on bundles in the dual solution, as a tradeoff is made between revenue and agent utility.

The Vickrey payments can be computed in the optimal dual solution that minimizes revenue, or in the *minimal competitive equilibrium* prices, when the *agents-are-substitutes* condition holds [BO99].

Minimal competitive equilibrium (CE) prices are defined as:

DEFINITION 6.2 [minimal CE prices] The minimal CE prices are computed in the optimal dual solution that *minimizes* the revenue to the auctioneer, π, or alternatively *maximizes* the marginal product (or utility) of the agents, $\sum_i p(i)$.

The *agents-are-substitutes* condition is stated in terms of agent marginal products:

DEFINITION 6.3 [agents-are-substitutes] Vickrey payments are computed at the minimal

competitive equilibrium prices if and only if

$$V(I) \Leftrightarrow V(K) \geq \sum_{j \in \mathcal{I} \setminus K} [V(I) \Leftrightarrow V(I \setminus j)] \quad \forall K \subseteq \mathcal{I}$$

where $V(K)$ denotes the value of the optimal solution to CAP with agents $K \subseteq \mathcal{I}$.

Intuitively, the marginal product of a group of agents is required to be greater than the sum marginal product of each agent in isolation. One can imagine factory workers, where each additional worker at the margin has a small effect on the productivity of the factory but the cumulative effect of having many people not turn up for work (a strike) is significant.

The equivalence between minimal CE prices and the Vickrey payment is easy to understand for a single item Vickrey auction. For a single item the price to the winning agent must be greater than the maximum price that any other agent is prepared to pay, i.e. the value of the second-highest bid, if the solution is to be in competitive equilibrium. The English auction selects this price by increasing prices by a minimal amount across rounds.

The minimal CE prices can be computed as a restricted dual problem, denoted [minCE], which selects the optimal dual solution that minimizes the auctioneer's revenue π:

$$\min_{p(i), p_i(S), \pi} \pi \qquad \text{[minCE]}$$

$$\text{s.t.} \quad p(i) + p_i(S) \geq v_i(S), \quad \forall i, S \qquad \text{(minCE-1)}$$

$$\pi \Leftrightarrow \sum_{[i,S] \in k} p_i(S) \geq 0, \quad \forall k \qquad \text{(minCE-2)}$$

$$\pi + \sum_i p(i) = V(I) \qquad \text{(minCE-3)}$$

$$p(i), p_i(S), \pi \geq 0, \quad \forall i, S$$

This restricted dual is different from the regular dual problem [DLP$_3$] in that the objective function is $\min \pi$ instead of $\min \pi + \sum_i p(i)$ and constraint (minCE-3) is new. Constraint (minCE-3) ensures that the value of the dual solution $p(i), \pi, p_i(S)$ computed by [minCE] is equal to the value of the optimal primal, and therefore an optimal dual solution. A similar approach was earlier proposed by Leonard [Leo83] to compute minimal CE prices and Vickrey payments in the assignment problem.

Bikchandani & Ostroy [BO99] prove the following result.

THEOREM 6.1 (min CE prices). *The minimal CE prices, as computed by [minCE], support the Vickrey payments to each agent when the agents-are-substitutes condition holds; i.e.* $p_i(S_i^*) = p_{\text{vick}}(i)$*, for minimal CE bundles prices* $p_i(S)$*.*

In the next section I develop a primal-dual formulation, which can compute the minimal CE prices without explicit information about either the value of the efficient allocation or agents' valuation functions.

6.1.1 A Primal-Dual Formulation

At this stage my approach departs from that in Bikchandani & Ostroy [BO99] and Bikchandani *et al.* [BdVSV01]. Bikchandani & Ostroy close their paper with a note that the existence of a linear program formulation for Vickrey payments might lead to an iterative Vickrey auction for CAP. Bikchandani *et al.* [BdVSV01] provide primal-dual auction interpretations of the Demange *et al.* [DGS86] and Gul & Stacchetti [GS00] auctions, noting that the price updates adjust prices on a "minimal overdemanded set" of items, which selects the optimal dual solution with minimal auctioneer revenue. Constructive methods to compute minimal dual solutions with primal-dual methods are also demonstrated for Ausubel [Aus97] and Ausubel [Aus00]. However, the authors make no constructive progress towards an iterative Vickrey mechanism for the general CAP.

I develop a primal-dual based formulation of [minCE] that does not require the value of $V(I)$. This value is not available in the competitive equilibrium at the end of an iterative auction. Similarly, the formulation should not require explicit information about agents' values $v_i(S)$ for bundles, beyond that which is available from best-response bids.

Given primal solution $x_i(S)$ and $y(k)$ and dual solution $p(i), \pi$, and $p_i(S)$, the complementary slackness conditions are:

$$x_i(S) > 0 \Rightarrow p(i) + p_i(S) = v_i(S) \tag{CS-1}$$

$$y(k) > 0 \Rightarrow \pi \Leftrightarrow \sum_{[i,S]\in k} p_i(S) = 0 \tag{CS-2}$$

$$p(i) > 0 \Rightarrow \sum_{S\subseteq\mathcal{G}} x_i(S) = 1 \tag{CS-3}$$

$$p_i(S) > 0 \Rightarrow \sum_{i\in\mathcal{I}} x_i(S) = \sum_{k\in K,[i,S]\in k} y(k) \tag{CS-4}$$

$$\pi > 0 \Rightarrow \sum_{k\in K} y(k) = 1 \tag{CS-5}$$

I formulate an alternative to restricted dual problem [minCE], which replaces constraint (minCE-3) with complementary-slackness conditions with the efficient allocation. It is safe to ignore conditions (CS-3), (CS-4) and (CS-5) because these are all trivially satisfied.

$$\min_{p(i),p_i(S),\pi} \pi$$

$$\text{s.t.} \quad p(i) + p_i(S) \geq v_i(S), \qquad \forall i, S \tag{D1}$$

$$\pi \Leftrightarrow \sum_{[i,S]\in k} p_i(S) \geq 0, \qquad \forall k \tag{D2}$$

$$p(i) + p_i(S_i^*) = v_i(S_i^*), \qquad \forall i \in I_{\text{sol}} \tag{CS1}$$

$$\pi \Leftrightarrow \sum_i p_i(S_i^*) = 0 \tag{CS2}$$

$$p(i), p_i(S), \pi \geq 0, \quad \forall i, S$$

Shorthand S_i^* is used to denoted the bundle allocated to agent i, i.e. the bundle corresponding with $x_i(S) = 1$, and $I_{\text{sol}} \subseteq \mathcal{I}$ is the subset of agents with a non-empty bundle, such that $i \in I_{\text{sol}} \Leftrightarrow \sum_S x_i(S) = 1$. It is straightforward to see that constraints (CS1) and (CS2) enforce complementary slackness with the optimal primal solution, while the other constraints are the standard constraints for dual feasibility.

Combining (CS1) and (D1), we require:

$$p(i) = v_i(S_i^*) \Leftrightarrow p_i(S_i^*) \geq \max_S (v_i(S) \Leftrightarrow p_i(S))$$

Combining (CS2) and (D2), we require:

$$\pi = \sum_i p_i(S_i^*) \geq \max_{k \in K} \sum_{[i,S] \in k} p_i(S)$$

With this the restricted dual [minCE] can be reformulated in terms of complementary-slackness conditions as [minCE-CS]:

$$\min_{p_i(S)} \sum_i p_i(S_i^*) \qquad \text{[minCE-CS]}$$

$$\text{s.t.} \quad v_i(S_i^*) \Leftrightarrow p_i(S_i^*) \geq \max_S (v_i(S) \Leftrightarrow p_i(S)) \qquad \text{(BR)}$$

$$\sum_i p_i(S_i^*) \geq \max_{k \in K} \sum_{[i,S] \in k} p_i(S) \qquad \text{(REV)}$$

$$p_i(S) \geq 0, \quad \forall i, S$$

In other words, every agent in the efficient allocation must continue to demand its bundle S_i^* at the adjusted prices (BR), and the revenue from the efficient allocation must continue to dominate the best possible revenue over all other allocations (REV).

This formulation will compute minimal CE prices without explicit information about the value of the efficient allocation. Although an agent's valuation function $v_i(S)$ still appears in the right-hand side of the (BR) constraints, this condition can be maintained by adjustment from CE prices; i.e., reduce the price to agent i on bundle S_i^* (the bundle it receives in the efficient allocation) by at least as much as the reduction in the price on all other bundles. The second condition (REV) only requires information about the prices, which is readily available to the auctioneer. I derive formulation [minCE-Adjust], which computes minimal CE prices from suitable CE prices and the efficient allocation in the next section.

The following result follows from Theorem 6.1.

LEMMA 6.1 *Linear program [minCE-CS] will compute the Vickrey payment to each agent when the agents-are-substitutes condition holds, as $p_i(S_i^*)$ where S_i^* is the bundle agent i receives in the efficient allocation and $p_i(S)$ are minimal CE prices.*

6.2 ADJUST: Discounts for Minimal CE Prices

In this section I propose a concrete algorithm ADJUST to solve [minCE-CS] without explicit information about agents' valuation functions, and without enumerating all (REV) constraints. ADJUST is used at the end of COMBAUCTION, and at the end of *i*Bundle, to adjust prices towards minimal CE prices and Vickrey payments after the auction terminates.

It is useful to introduce the concept of prices that are a *negative translation* of an agent's valuation function, written $p_i(\cdot) = v_i(\cdot) \ominus C$.

DEFINITION 6.4 [negative translation] Vector $\mathbf{x}$ is a negative translation of $\mathbf{y}$ by $C \geq 0$, written $\mathbf{x} = \mathbf{y} \ominus C$ if $x_j = \max(0, y_j \Leftrightarrow C)$ for all $j \in \{1, \ldots, \dim(x)\}$.

Given that the objective in [minCE-CS] is to minimize $\sum_i p_i(S_i^*)$, while constraint (REV) requires this sum to be greater than the maximal revenue over all allocations, the problem can be solved by setting values on $p_i(S_i^*)$ and computing values for the other prices that are as *small as possible* without violating (BR).

For an agent i in the final allocation, with price $p_i(S_i^*)$ on bundle S_i^*, the smallest price on bundle $S_i \neq S_i^*$ that also satisfies (BR) is:

$$p_i(S_i) = \max(0, v_i(S) \Leftrightarrow v_i(S_i^*) + p_i(S_i^*)), \quad \text{for all } S \subseteq \mathcal{G}$$

For an agent i not in the final allocation we require

$$p_i(S) = v_i(S), \quad \text{for all } S \subseteq \mathcal{G}$$

Prices that are negative translations of agent valuation functions, i.e. $p_i = v_i \Leftrightarrow C$, for $C = v_i(S_i^*) \Leftrightarrow p_i(S_i^*)$, satisfy these conditions. The restricted dual can now be restated with only as many decision variables, $C_1, \ldots, C_I$, as there are agents:

$$
\begin{aligned}
&\max_{(C_1,\ldots,C_I)} \sum_i C_i \\
\text{s.t.} \quad & p_i(S) = v_i(S) \ominus C_i \quad \forall i, S \\
& C_i = 0 \quad \forall i \notin I_{\text{sol}} \\
& p_i(S_i^*) \Leftrightarrow C_i \geq 0 \quad \forall i \qquad (*) \\
& \sum_i p_i(S_i^*) \geq \max_{k \in K} \sum_{[i,S] \in k} p_i(S) \\
& C_i \geq 0, \quad \forall i
\end{aligned}
$$

In words, we compute minimal CE prices as the maximal set of discounts C_i from agent valuation functions that maintain complementary slackness conditions with the efficient allocation.

The final adjusted prices $p_i(S_i^*) \Leftrightarrow C_i$ are non-negative by constraint (*). The negative translation, $p_i(S) = v_i(S) \ominus C_i$, can be expressed with the following additional constraints:

$$
\begin{aligned}
& p_i(S_i^*) \leq v_i(S_i^*) \Leftrightarrow C_i \quad \forall i \in I_{\text{sol}} \\
& p_i(S) \geq v_i(S) \Leftrightarrow C_i \quad \forall (i,S) \text{ s.t. } x_i(S) = 0 \\
& p_i(S) \geq 0 \quad \forall i, S
\end{aligned}
$$

Dropping information about agent valuation functions, we can rewrite this in terms of prices $p_i^T(S)$ that are *competitive equilibrium* prices *and negative translations* of agents' valuation functions.

Linear program [minCE-Adjust] computes discounts $\Delta_i \geq 0$ to each agent i, to adjust prices from $p_i^T(S)$ to $p_i^T(S) \Leftrightarrow \Delta_i$:

$$\max_{(\Delta_1,\ldots,\Delta_I)} \sum_i \Delta_i \qquad \text{[minCE-Adjust]}$$

$$\text{s.t.} \quad p_i(S) = p_i^T(S) \ominus \Delta_i \quad \forall i, S \qquad \text{(BR)}$$

$$\Delta_i = 0 \quad \forall i \notin I_{\text{sol}}$$

$$p_i^T(S_i^*) \Leftrightarrow \Delta_i \geq 0$$

$$\sum_i p_i(S_i^*) \geq \max_{k \in K} \sum_{[i,S] \in k} p_i(S) \qquad \text{(REV)}$$

$$\Delta_i \geq 0, \quad \forall i$$

Formulation [minCE-Adjust] is useful because the prices computed at the end of CombAuction(3), and equivalently at the end of iBundle(3) with myopic best-response agent strategies, are CE prices *and* negative translations of an agent valuation functions.

6.2.1 An Efficient Implementation

The following lemma allows a solution to be computed without enumerating all $k \in K$ and checking the (REV) constraints explicitly.

Given discounts $\Delta = (\Delta_1, \ldots, \Delta_I)$, i.e. such that $p_i(S) = p_i^T(S) \ominus \Delta_i$, let P_Δ^* denote the value of the revenue-maximizing allocation, and $(P_{\Delta,-i})^*$ denote the value of the revenue-maximizing allocation without agent i.

Lemma 6.2 *The maximal discount, Δ_i, to agent i without breaking (REV) is* $\Delta_i = \min(p_i^T(S_i^*) \Leftrightarrow \Delta, (P_\Delta^* \Leftrightarrow (P_{\Delta,-i})^*))$.

Proof. A simple feasibility argument shows that the only allocations that might provide more revenue to the auctioneer as prices are decreased on bundles to agent i are allocations that do not assign a bundle to agent i. Any allocation that still contains agent i would have also provided more revenue before prices are reduced. Therefore, prices can be reduced until the adjusted revenue from allocation S^* equals the best alternative, i.e. $P_\Delta^* \Leftrightarrow \Delta_i = (P_{\Delta,-i})^*$. ∎

This suggests the Adjust algorithm, in Figure 6.1. Adjust computes agent discounts, taking each agent in S^* in turn. Price reductions to each agent in the allocation are

ADJUST: input $p_i^T(\cdot), S^*$, output Δ
$P = P^*$;
for each $i \in I^*$ {
    $\Delta_i = \min\{P \Leftrightarrow (P_{\Delta,-i})^*, p_i^T(S_i^*)\}$;
    $P = P \Leftrightarrow \Delta_i$;
};

Figure 6.1: The ADJUST algorithm

considered incrementally and not independently; discounts already allocated to agents $i < j$ are considered when computing a discount for agent j. This is relaxed in ADJUST*, introduced in the next section.

The following result is immediate:

LEMMA 6.3 *Procedure* ADJUST *computes minimal CE prices from CE prices p^T that are negative translations of agent valuation functions.*

Although the optimization problem in each round, to compute the revenue maximizing allocation given current discount, $(P_{\Delta,-i})^*$, is NP-hard (equivalent to the CAP), and therefore worst-case exponential, search-based algorithms have been demonstrated to perform well on average [San99, FLBS99, ATY00]. The advantage of ADJUST over a direct linear program implementation of [minCE-Adjust] is that it is not necessary to explicitly enumerate all constraints (REV). Essentially, ADJUST computes a solution to the linear program [minCE-Adjust] through *implicit* enumeration, pruning many possible allocations from consideration.

A fast approximate method to compute the solution to ADJUST without solving the revenue-maximization problem $(P_{\Delta,-i})^*$ is suggested in the next section. The ADJ-PIVOT method uses computation performed during the auction to compute the minimal CE prices.

Combining Lemma 6.3 with the agents-are-substitutes condition, we have the following important result:

THEOREM 6.2 (Vickrey payments). *Procedure* ADJUST *computes Vickrey payments when the agents-are-substitutes condition holds and prices $p^T(\cdot)$ are CE prices and negative translations of agent valuation functions.*

The condition in Lemma 6.3 on prices is sufficient but not necessary. A weaker *necessary and sufficient* condition, ϵ-CS1-tightness is:

DEFINITION 6.5 [ϵ-CS1-tightness] (i) For every agent j in the optimal allocation all bundles it receives in any second best allocation are in its best-response set at prices $p^T(\cdot)$, (ii) For every agent j not in the optimal allocation, its price must be within ϵ if its value for any bundle it receives in a second-best allocation.

The ϵ-CS1-tightness condition implies that it is not possible to reduce the price on any bundle that is in a binding second-best revenue-maximizing solution without violating the best-response complementary slackness condition, CS1.

LEMMA 6.4 *Procedure* ADJUST *computes minimal competitive equilibrium prices from CE prices $p^T(\cdot)$ if and only if agent best-response bids satisfy ϵ-CS1-tightness, as bid increment $\epsilon \to 0$.*

In the next section I use ADJUST at the end of COMBAUCTION(3) to compute minimal CE prices from best-response agent information.

6.2.2 A Primal-Dual Algorithm to Compute Minimal CE Prices

Recall that COMBAUCTION(3) refers to the variant of the primal-dual algorithm for the third-order linear program formulations, [LP_3] and [DLP_3], in which separate prices $p_i(S)$ are maintained for each agent in every iteration. Also, the final prices satisfy (BR) and (REV), and satisfy the price-indifference property because of agent best-response bidding strategies and the price-update rules.

From Lemma 6.3 we can state the following result:

THEOREM 6.3 (min CE). COMBAUCTION(3) *followed by* ADJUST *computes minimal CE prices and the efficient allocation, as the bid increment $\epsilon \to 0$.*

From Theorem 6.2 we can also make a claim about the ability to compute Vickrey payments with COMBAUCTION(3) followed by ADJUST:

THEOREM 6.4 (vickrey). COMBAUCTION(3) *followed by* ADJUST *computes Vickrey*

payments and the efficient allocation when the agents-are-substitutes condition holds, as the bid increment $\epsilon \to 0$.

This iterative procedure captures and extends all known previous methods to compute Vickrey payments from sequential agent best-response bids (see Table 4.7) in the combinatorial allocation problem, including for Gross-substitutues agent preferences.

6.2.3 Speeding-Up: Pivot Allocations

Procedure ADJUST is NP-hard because it computes the value of the revenue-maximizing allocation once with all agents, and then for the system with each agent removed.

In Parkes & Ungar [PU00b] we propose a fast approximate method, ADJ-PIVOT to compute minimal competitive equilibrium prices. The algorithm uses computation already performed during earlier rounds of the auction to approximate the value of second-best allocations. Section 7.7 in the next chapter demonstrates that the method is very accurate and very fast. The experimental success of the method provides useful insight into the nature of minimal competitive equilibrium prices.

To compute the revenue-maximizing allocation $(P_{\Delta,-i})^*$ without agent i, ADJ-PIVOT computes the value of the best allocation over all pivotal allocations.

DEFINITION 6.6 [pivotal allocations] The *pivotal* allocations are the set of partitions that have formed provisional allocations in one or more iterations of COMBAUCTION.

It is perhaps reasonable that these pivot allocations are the allocations likely to represent allocations with high value at the final prices.

The approximate method to compute minimal CE prices from pivotal allocations is formulated and solved as a linear program:

$$\max_{(\Delta_1,\ldots,\Delta_I)} \sum_i \Delta_i \qquad \text{[minCE-Pivot]}$$

$$\text{s.t.} \quad p_i(S) = p_i^T(S) \ominus \Delta_i \quad \forall i, S$$

$$\Delta_i = 0 \quad \forall i \notin I_{\text{sol}}$$

$$p_i^T(S_i^*) \Leftrightarrow \Delta_i \geq 0$$

$$\sum_i p_i(S_i^*) \geq \max_{k \in Pivot} \sum_{[i,S] \in k} p_i(S) \qquad (*)$$

$$\Delta_i \geq 0, \quad \forall i$$

where the right-hand side of (*) computes the best allocation consistent with the set *Pivot* of pivotal allocations. An individual pivotal allocation, $k \in Pivot$, defines an allocation of bundles to agents. When matching against the pivot set in ADJ-PIVOT it is useful to allow matches against *permutations* of pivot allocations, where the particular set of agents that receive bundles in a partition can be different from that in an intermediate round of the auction.

Experimental results in the next chapter show that ADJ-PIVOT is very fast, and appears as accurate as ADJUST for small bid increments. The experimental success of ADJ-PIVOT provides an intuitive understanding of minimal CE prices:

PROPOSITION 6.1 *Minimal competitive equilibrium prices, and often Vickrey payments, are approximately the smallest prices that the agents in the final "winning coalition" had to bid with hindsight to beat the best provisional allocation that dropped at least one agent in the coalition and included at least one outside agent.*

Essentially the price adjust method ADJ-PIVOT allows agents that are in the winning allocation and important provisional allocations to pay less than an agent that is very dependent on the winning coalition to receive a bundle.

6.3 ADJUST*: Discounts for Vickrey Payments

The key to VICKAUCTION is to recognize that although there are problems for which the agents-are-substitutes condition does not hold, the Vickrey payment for any agent can

always be computed as the minimal price for its bundle in the efficient allocation over *all* competitive equilibrium prices.

Recall that the minimal CE prices are prices in the optimal dual solution that minimize the revenue to the auctioneer, or equivalently maximize the sum marginal product over all agents. In fact, min CE prices compute Vickrey payments when there is a *unique* set of minimal CE prices (this is an alternative interpretation of the agents-are-substitutes condition):

LEMMA 6.5 *If the agents-are-substitutes condition holds there is a unique set of minimal CE prices (in terms of the prices for bundles in the allocation), and those prices equal Vickrey payments.*

PROOF. (sketch) All agents in the efficient allocation remain in the efficient allocation without any one agent, by the agents-are-substitutes condition. It follows that it is the values to *the agents not in the efficient allocation* that define the price discounts to each agent, and the order with which agents are selected in ADJUST does not change their discount. This implies that there is a unique set of minimal CE prices. ∎

This generalizes in the following useful way:

LEMMA 6.6 *If the agents-are-substitutes condition does not hold there are multiple minimal CE prices (in terms of the prices for bundles in the allocation), and the Vickrey payment for any agent i is computed as the minimal price on its bundle in the efficient allocation over all minimal CE prices.*

To understand why this is the case, consider the following linear program, [VDLP(i)], which maximizes the utility, $p(i)$, to agent i (or alternatively minimizes the price agent i pays for its optimal bundle S_i^*). This linear program is the *restricted Vickrey dual* for agent i.

$$\max_{p(i),p_i(S),\pi} p(i) \qquad \text{[VDLP}(i)\text{]}$$

$$\text{s.t.} \quad p(i) + p_i(S) \geq v_i(S), \quad \forall i, S \qquad \text{(VDLP-1)}$$

$$\pi \Leftrightarrow \sum_{[i,S]\in k} p_i(S) \geq 0, \quad \forall k \qquad \text{(VDLP-2)}$$

$$\pi + \sum_i p(i) = V(I) \qquad \text{(VDLP-3)}$$

$$p(i), p_i(S), \pi \geq 0, \quad \forall i, S$$

I prove that [VDLP(i)] computes agent i's marginal product, $p(i) = V(I) \Leftrightarrow V(I \setminus i)$, and sets $p_i(S_i^*) = p_{\text{vick}}(i)$, agent i's Vickrey payment:

LEMMA 6.7 (Vickrey dual problem). *The solution to the restricted Vickrey dual for agent i computes $p_i(S_i^*) = p_{\text{vick}}(i)$ in all problems, where S_i^* is the bundle agent i receives in the efficient allocation.*

In other words, the Vickrey payment, $p_{\text{vick}}(i)$, for agent i is computed in the optimal dual solution that minimizes the dual price, $p_i(S_i^*)$, on the bundle, S_i^*, that agent i receives in the efficient allocation.

PROOF.

The proof is constructive, based on Lemma 6.3 and an inspection of the ADJUST procedure. Lemma 6.3 states that ADJUST computes minimal competitive equilibrium prices for input prices $p^T(\cdot)$ that: (a) are negative translations of agent valuation functions; (b) satisfy (BR), such that agent i's optimal allocation is in its best-response set; and (c) satisfy (REV), such that the auctioneer maximizes revenue at the prices with the value-maximizing allocation.

Consider prices $p_i(S) = v_i(S)$ for all agents $i \in \mathcal{I}$, and all bundles $S \subseteq \mathcal{G}$. Clearly these prices satisfy conditions (a), (b) and (c). To complete the proof notices that agents can be selected in *any order* in ADJUST, and we will still compute a valid set of minimal CE prices. Without loss of generality, suppose agent i is selected first. The discount Δ_i to

agent i is computed as:

$$\begin{aligned}\Delta_i &= \min\{P \Leftrightarrow (P_{\Delta,-i})^*, p_i^T(S_i^*)\} \\ &= \min\{V^* \Leftrightarrow (V_{-i})^*, v_i(S_i^*)\} \\ &= V^* \Leftrightarrow (V_{-i})^*\end{aligned}$$

and agent i's adjusted price, the price for its bundle in this optimal dual solution, is:

$$\begin{aligned}p_i &= p_i(S_i^*) \Leftrightarrow \Delta_i \\ &= v_i(S_i^*) \Leftrightarrow (V^* \Leftrightarrow (V_{-i})^*) \\ &= p_{\text{vick}}(i)\end{aligned}$$

which is agent i's Vickrey payment. Therefore, because any agent can be selected first, and because there is always a set of minimal CE prices consistent with the adjusted price computed for the initial agent, the Vickrey payment to any agent i can be computed as the minimal price over all CE prices. Note— Bikchandani & Ostroy [BO99] earlier show that Vickrey payments are always a *lower bound* on all minimal CE prices. ▮

The significance of this result is that it is always possible to compute Vickrey payments to each agent with "enough" primal dual information, solving $|\mathcal{I}|$ restricted Vickrey dual problems— one to maximize the marginal product for each agent.

We can state the important following result:

THEOREM 6.5 *The Vickrey payment for agent i can be computed as the minimal price for bundle S_i^* in the efficient allocation over all minimal CE prices, i.e. over all optimal dual solutions that minimize the auctioneer's revenue.*

The obvious next step is to propose a slightly modified price-adjustment procedure, ADJUST*, to solve the restricted Vickrey dual [VDLP(i)] for all agents. See Figure 6.2.

ADJUST* is identical to ADJUST except that it computes the discount to each agent separately, and independently of the discounts assigned to other agents. Whenever $p_i(\cdot) = v_i(\cdot)$ we compute the Vickrey discount for each agent. In other cases, the discounted price remains a competitive equilibrium price in some dual solution, and is no less than the agent's Vickrey payment.

The next lemma follows immediately from Lemma 6.3.

ADJUST*: `input` $p_i^T(\cdot), S^*$, `output` Δ
$P = P^*$;
`for each` $i \in I^*$ {
    $\Delta_i = \min\{P \Leftrightarrow (P_{-i})^*, p_i^T(S_i^*)\}$;
};

Figure 6.2: Procedure ADJUST*.

LEMMA 6.8 ADJUST* *computes a discounted price* $p_i^T(S_i^*) \Leftrightarrow \Delta_i$ *to agent* i *that is no less than its Vickrey payment for prices* $p^T(\cdot)$ *that are negative translations of agents' valuation functions, and also satisfy (BR) and (REV).*

We also have the following result, from Lemma 6.7.

LEMMA 6.9 ADJUST* *computes a discounted price equal to agent* i*'s Vickrey payment for prices that are negative translations of agents' valuation functions, satisfy (BR) and (REV), as the prices approach agents' valuation functions.*

Let us characterize the conditions that enable us to compute Vickrey payments; i.e. how far away can we be from complete information revelation but still compute the outcome of the GVA? Remember, we want to compute Vickrey payments without complete information about agent valuation functions.

As before, consider initial prices $p_i^T(\cdot)$ that are negative translations of agent valuation functions, and satisfy (BR) and (REV) at the optimal allocation S^*. Linear program [VDLP-CS(i)] is a complementary slackness formulation of the restricted Vickrey dual [VDLP(i)] that does not require explicit information about $V(I)$ and $v_i(\cdot)$.

$$\max_{(\Delta_1,\cdots,\Delta_I)} \Delta_i \qquad \text{[VDLP-CS}(i)\text{]}$$

$$\text{s.t.} \quad p_i(S) = p_i^T(S) \ominus \Delta_i \quad \forall i, S \qquad \text{(BR)}$$

$$\Delta_i = 0 \quad \forall i \notin I_{\text{sol}}$$

$$p_i^T(S_i^*) \Leftrightarrow \Delta_i \geq 0$$

$$\sum_i p_i(S_i^*) \geq \max_{k \in K} \sum_{[i,S] \in k} p_i(S) \qquad \text{(REV)}$$

$$\Delta_i \geq 0, \quad \forall i$$

In words, [VDLP-CS(i)], computes discounts to each agent to maximize the individual discount, Δ_i, to agent i from its CE price, while maintaining complementary slackness conditions with the efficient allocation. Procedure ADJUST* solves this linear program. Prices $p_i^T(\cdot)$ are negative translations of agents' valuation functions, i.e. $p_i^T(S) = v_i(S) \ominus C_i$, for some $C_i \geq 0$. In proving Lemma 6.7 we showed that $C_i = 0$ for all agents i is a sufficient condition. The necessary and sufficient conditions on the initial discounts $C_i \geq 0$ that define agents' initial prices, for [VDLP-CS(i)] to compute the Vickrey payment to agent i, are:

(a) For agents $j \notin I_{\text{sol}}$. We already require $C_j \leq 0$ by myopic best-response and (CS-1). Now, if agent j receives a bundle in the second-best allocation without agent i we need either $C_j = 0$, or a zero discounted price for agent i.

(b) For agents $j \in I_{\text{sol}}$. We already require $C_j \geq 0$ by myopic best-response and (CS-1). Now, if agent j does not receive a bundle in the second-best allocation without agent i we need either $C_j = 0$, or a zero discounted price for agent i.

Putting this all together, and ignoring this additional clause, we can state *joint sufficient* conditions on prices $p_i^T(S)$ for procedure ADJUST* to compute Vickrey payments to every agent.

THEOREM 6.6 (ADJUST* optimality). *Procedure* ADJUST* *computes Vickrey payments from CE prices $p_i^T(\cdot)$ that are (a) negative translations from agent valuation functions; and (b) equal to value $v_i(\cdot)$ for every agent i in the optimal allocation but not in a second-best allocation for some agent $j \neq i$, $j \in I_{\text{sol}}$.*

where the second-best allocation without agent i is the allocation that maximizes revenue without agent i at the prices, i.e. the value of $(P_{-i})^*$. Condition (b) extends what was required in Lemma 6.3 for procedure ADJUST to compute minimal CE prices.

It is useful to define a *Vickrey Test*, in terms of intuitive descriptions of the state of a competitive equilibrium solution, including the *dependents* of an agent, the *active* agents, and a *needy* agent.

Let P^* denote the value of the revenue maximizing allocation at current prices, with $W^* \subseteq \mathcal{I}$ the set of agents in that allocation. Also, let $(P_{-i})^*$ denote the value of the revenue maximizing allocation without agent i at the current prices, the *second-best allocation*, and $(W_{-i})^*$ denote the agents in that allocation.

The dependents of agent i are used to check whether ADJUST* computes Vickrey payments at current prices.

DEFINITION 6.7 [dependent agents] The dependents $\alpha(i)$ of agent i are:

$\alpha(i) = W^* \setminus ((W_{-i})^* \bigcup i)$, if $i \in W^*$

$\alpha(i) = \emptyset$, otherwise.

The dependents of agent i are agents that receive a bundle in allocation $\mathbf{S}^*$ but do not receive a bundle in the second-best allocation without agent i at the current prices.

An *active* agent is an agent that still requests bundles at the current prices:

DEFINITION 6.8 [active agents] Agent i is active at prices $p_i(\cdot)$ if its best-response set of bundles (those bundles that maximize its utility) is non-empty, i.e. the agent continues to demand one or more bundles.

A *needy* agent is an agent in the efficient allocation that still has a non-zero adjusted price in the algorithm:

DEFINITION 6.9 [needy] An agent i is needy if its adjusted price is non-zero.

With this, I restate the sufficient conditions in Theorem 6.6 for Vickrey payments as the *Vickrey test*:

DEFINITION 6.10 [Vickrey test] CE prices satisfy the Vickrey test condition when they are negative translations of agents' valuation functions, and no *needy* agent in the efficient allocation has any *active dependent* agents.

Procedure ADJUST* computes Vickrey payments from CE prices whenever the prices satisfy the Vickrey test. Note that the auctioneer can compute the *needy*, *active* and

dependent agents from best-response bids.

Finally, we can define necessary and sufficient conditions for Vickrey payments, in terms of the ϵ-CS1-tightness condition introduced in Section 6.2:

PROPOSITION 6.2 *Procedure* ADJUST* *computes Vickrey payments from CE prices* $p_i^T(\cdot)$ *if and only if best-response bids satisfy* ϵ*-CS1-tightness;* and *no* needy *agent* i *in the efficient allocation has any* active dependent *agents.*

In the following section I give an illustrative example of ADJUST*.

6.3.1 Example

Consider Problem 7 in Table 6.1. The optimal allocation is $\mathbf{S}^* = (A, B, \emptyset)$, i.e. with items are allocated to agents 1 and 2. The Vickrey prices are $p_{\text{vick}}(1) = 30 \Leftrightarrow (70 \Leftrightarrow 40) = 0$ and $p_{\text{vick}}(2) = 40 \Leftrightarrow (70 \Leftrightarrow 50) = 20$.

	A	B	AB
Agent 1	30*	0	30
Agent 2	0	40*	40
Agent 3	0	20	40

Table 6.1: Problem 7. Efficient allocation indicated *. Vickrey payments: $p_{\text{vick}}(1) = 30 - (70 - 40) = 0$, $p_{\text{vick}}(2) = 40 - (70 - 50) = 20$.

First, let us check the "agents as substitutes" condition, to determine whether Vickrey payments are supported in competitive equilibrium. The marginal products are $\text{MP}(1) = V(I) \Leftrightarrow V(I \setminus 1) = 70 \Leftrightarrow 40 = 30$, $\text{MP}(2) = 70 \Leftrightarrow 50 = 20$, and $\text{MP}(3) = 70 \Leftrightarrow 70 = 0$. We must check:

$$V(I) \Leftrightarrow V(K) \geq \sum_{i \in (\mathcal{I} \setminus K)} \left[V(I) \Leftrightarrow V(I \setminus i) \right], \quad \forall K \subseteq \mathcal{I}$$

This condition does not hold, because for $K = \{3\}$, we have $V(123) \Leftrightarrow V(3) < \text{MP}(1) + \text{MP}(2)$, because $70 \Leftrightarrow 40 < 30 + 20$. Therefore the Vickrey payments are not supported in competitive equilibrium. Indeed we might suspect this because $p_{\text{vick}}(1) + p_{\text{vick}}(2) < v_3(AB)$.

Consider using ADJUST and ADJUST* in two scenarios. Prices are in competitive equilibrium in both cases, and are negative invariants of agent valuation functions. By Lemma 6.3 we expect to compute minimal CE prices with ADJUST in both scenarios.

Vickrey payments will depend on the conditions of Theorem 6.6, i.e. on the prices and second-best allocations of agents 1 and 2.

- **Scenario 1.** Prices are $p_1^T = \{25, 0, 25\}$, $p_2^T = \{0, 25, 25\}$ and $p_3^T = \{0, 20, 40\}$. ADJUST computes discounts $\Delta_1 = 50 \Leftrightarrow 40 = 10$ and $\Delta_2 = 40 \Leftrightarrow 40 = 0$, *or* $\Delta_2 = 50 \Leftrightarrow 45 = 5$ and $\Delta_1 = 45 \Leftrightarrow 40 = 5$, depending on which agent is selected first. The adjusted, minimal CE prices are $p_1 = \{15, 0, 15\}$ and $p_2 = \{0, 25, 25\}$, *or* $p_1 = \{20, 0, 20\}$ and $p_2 = \{0, 20, 20\}$, with $p_3 = \{0, 20, 40\}$ in both cases. ADJUST* computes $\Delta_1 = 50 \Leftrightarrow 40 = 10$ and $\Delta_2 = 50 \Leftrightarrow 45 = 5$, for final adjusted prices $p_1(A) = 15$ and $p_2(B) = 20$. Agent 2 gets its Vickrey payment because agent 1 is in the second-best allocation without agent 2. However, agent 1 does not gets its Vickrey payment because agent 2 is *not* in the second-best allocation without agent 1, and does not bid its full value for B.

- **Scenario 2.** Prices are $p_1^T = \{25, 0, 25\}$ as before, but now $p_2 = \{0, 40, 40\}$, i.e. equal to its valuation function. We know immediately that ADJUST* now computes the Vickrey payments because the Vickrey Test condition is satisfied for both agents 1 (it is not a dependent agent of 2), and agent 2 (it bids its valuation function). The discounts computed in ADJUST* are $\Delta_1 = 65 \Leftrightarrow 40 = 25$ and $\Delta_2 = 65 \Leftrightarrow 45 = 20$, for adjusted prices $p_1(A) = 25 \Leftrightarrow 25 = 0 = p_{\text{vick}}(1)$ and $p_2(B) = 40 \Leftrightarrow 20 = 20 = p_{\text{vick}}(2)$. Notice that agent 1 does not need to reveal complete information about its valuation function. Procedure ADJUST computes either $\Delta = (25, 0, 0)$ *or* $\Delta = (5, 20, 0)$, which give minimal CE prices but not Vickrey payments to every agent.

This example indicates how it is possible to compute (and verify) Vickrey payments from CE prices, even when Vickrey payments are not supported in any single set of prices.

6.4 VICKAUCTION: A Primal-Dual Vickrey Algorithm

In this section I present a primal dual algorithm, VICKAUCTION, that provably computes Vickrey payments and the efficient allocation with best-response agent bids. In the next chapter I develop an extended *i*Bundle auction, *i*Bundle Extend&Adjust, which is designed to implement VICKAUCTION as an ascending-price auction.

VickAuction is a sequential composition of CombAuction(3), as described in Chapter 4, with a new algorithm PhaseII:

$$\text{VickAuction} = \text{CombAuction}(3) \cdot \text{PhaseII}$$

VickAuction has a natural interpretation as an ascending-price combinatorial auction because it has the following features:

(1) the only interaction with agents is via myopic best-response requests;

(2) prices are ascending throughout the algorithm;

(3) there is always a feasible allocation, the *provisional allocation* in an auction.

CombAuction(3) terminates with the efficient allocation S^* and competitive equilibrium prices $p_i^T(S)$ that are negative translations of agent valuation functions.

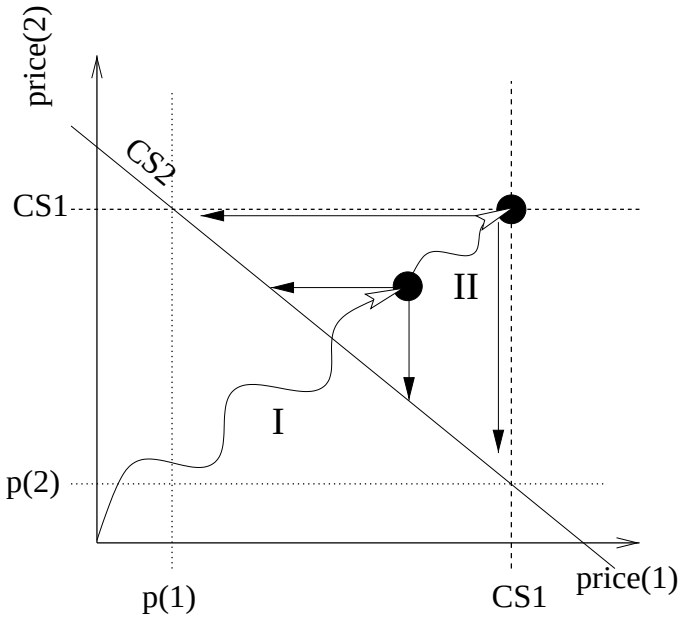


Figure 6.3: PhaseII: Collecting additional primal-dual information to compute Vickrey payments

Figure 6.3 illustrates the role of PhaseII. The Figure plots the complementary-slackness constraints with the efficient allocation in a simple problem with only two agents in the efficient allocation. On the x-axis is the price on the bundle that agent 1 receives in the efficient allocation. On the y-axis is the price on the bundle that agent 2 receives in the efficient allocation. To the left of vertical line CS1 the prices satisfy CS1 for agent 1 (i.e. the price is less than its value). Below line horizontal line CS1 the prices satisfy CS1 for agent 2 (i.e. the price is less than its value). To the upper-right side of diagonal line

CS2 the prices satisfy CS2 (i.e. the efficient allocation continues to maximize revenue for the auctioneer). Inside the triangle, between these constraints, prices are all optimal dual (or competitive equilibrium) prices on the bundles. The minimal CE price to agent 2, indicated $p(2)$ is its Vickrey payment. The minimal CE price to agent 1, indicated $p(1)$ is its Vickrey payment.

At the end of CombAuction(3), or Phase I in VickAuction, the primal-dual algorithm might be in the state indicated by the first solid circle, closest to the origin. At this point Adjust* will adjust prices to the points shown, but there is not enough information to compute minimal prices over the simplex of optimal dual prices. The purpose of PhaseII is to get enough additional information to move to a point like the second solid circle, from which Adjust* drops down to the two extremal values and computes Vickrey payments. In this problem agents 1 and 2 both needed to reveal their complete value for their bundle in the efficient allocation, but this is not the case in general. The path from I to II on the plot represents the "valuation (or information) cost of achieving incentive-compatibility".

Module PhaseII continues to increase prices and request best-response information from agents until the prices satisfy the *Vickrey test* conditions. As soon as this condition holds there is enough primal-dual information to adjust prices to Vickrey payments. PhaseII is described in Figure 6.4.

The combined algorithm, CombAuction(3) · PhaseII, which we call VickAuction, is optimal.

Theorem 6.7 (VickAuction optimality). VickAuction *is a primal-dual algorithm to compute the efficient allocation and the Vickrey payments for the combinatorial allocation problem.*

Proof. CombAuction(3) computes competitive equilibrium prices $p_i^*(\cdot)$ that are negative translations of agent valuation functions and the efficient allocation S^*. By Lemma 6.10 the revenue-maximizing allocation in every round of PhaseII remains allocation S^* computed at the end of CombAuction(3). Prices remain negative translations of agent valuation functions, and CE prices, because the prices continue to be increased on the basis of best-response bids from agents. PhaseII terminates with the Vickrey test conditions (definition 6.10) because there are no *needy* agents that also have *active*

PHASEII
Input: prices $p_i^T(S)$ and allocation S^* from COMBAUCTION
active = active(BR(p^T));
compute dependents(i) = $W^* \Leftrightarrow (W_{-i})^* \Leftrightarrow i$ for all i;
Δ_{init} = ADJUST*(p_i^T, S^*);
needy(i) = true if $\Delta_{\text{init}}(i) < p_i^T(S_i^*)$, false otherwise;
$\Delta_i = \Delta_{\text{init}}(i)$; $p_i(\cdot) = p_i^T(\cdot)$;
while ($\exists i$: needy(i) & (dependents(i) $\cap$ active $\neq \emptyset$)) {
  for $i \in$ active
    if ($\exists j$: needy(j) & $i \in$ dependents(j))
      $p_i(\cdot)$ = nonanon_update(BR_i, p_i);
  compute best-response set $\text{BR}_i(\mathbf{p}_i)$ for every agent i;
  active = active(BR(p));
  $\Delta_i = \Delta_i + |\text{dependents}(i) \cap \text{active}|\ \epsilon$;
  if ($\Delta_i \geq p_i^T(S_i^*)$)
    needy(i) = false;
  else {
    compute $(W_{-i})^*$;
    compute dependents(i) = $W^* \Leftrightarrow (W_{-i})^* \Leftrightarrow i$;
  };
}
output: adjusted prices $\max(0, p_i^T(S_i^*) \Leftrightarrow \Delta_i)$ to each agent.

Figure 6.4: The PHASEII algorithm.

dependents.

Instead of computing the adjusted price with ADJUST* explicitly, the discount $\Delta(i)$ is initialized explicitly with ADJUST* at the start of PHASEII and then maintained across rounds. The difference between P^* and $(P_{-i})^*$ for agent i increases in each round by the number of *active dependents* of agent i, multiplied by the minimal bid increment, plus an additional ϵ if the agent's own prices also increase; the effect is to increase the discount to agent i by $|\texttt{dependents(i)} \cap \texttt{active}|\ \epsilon$ with respect to its price *at the start of* PHASEII.

The final adjusted price $\max(0, p_i^T(S_i^*) \Leftrightarrow \Delta(i))$ for every agent i is equal to its Vickrey payment as $\epsilon \to 0$. ∎

The proof uses the following lemma.

LEMMA 6.10 *The revenue-maximizing allocation S^* does not change as prices are increased in* PHASEII. *The value of the revenue-maximizing allocation therefore increases by ϵ every time the price is increased to one of the agents in S^*.*

PROOF. No allocation to only agents in W^*, i.e. agents currently in the provisional allocation, can have more value than the current allocation as prices change because the price on bundles S_i^* increases by at least as much as the price on all other bundles to agent i (because of myopic best-response and the price-update rule). A simple feasibility argument shows that no solution that combines different bundles to agents in W^* with some bundles to agents outside of W^* can become value-maximizing without the same allocation having more value than solution S^* at the original prices. Of course, the prices are not increased to any agent not in W^*. ∎

6.4.1 Speeding-up: Pivot Allocations

Just as ADJ-PIVOT is a fast but accurate method to compute minimal CE prices at the end of COMBAUCTION, a similar technique ADJ-PIVOT* can be used to speed-up ADJUST* and VICKAUCTION.

The inner-loop of ADJUST* computes the value of the second-best allocation $(P_{-i})^*$ at the current prices (see Figure 6.2). ADJ-PIVOT* computes this value with the following approximation:

$$(P_{-i})^* \approx \max_{x \in Pivot} \sum_{[j,S] \in x, j \neq i} p_j^T(S)$$

where *Pivot* is the set of provisional allocations computed during CombAuction, and a single pivotal allocation $x \in Pivot$ defines an allocation $[i, S] \in x$ of bundles to agents. The approximation restricts attention to allocations that have proved to be interesting during the auction, and avoids computing a new solution to a large combinatorial optimization problem in each round.

The same method can be used to approximate the computation of $(W_{-i})^*$, i.e. the agents in the second-best allocation without agent i, during PhaseII of VickAuction, also computing the approximation based on pivot allocations computed during the CombAuction phase of the VickAuction algorithm.

Chapter 7

*i*Bundle Extend & Adjust

Much of my dissertation addresses a fundamental problem with the GVA, which is that it requires agents to compute and reveal their values for all combinations of items. This can be very difficult for bounded-rational agents with limited or costly computation. The complete information requirement arises because of the *single-shot* nature of the auction. Without an option to ask an agent for more information a mechanism can only compute the efficient allocation in every problem instance with complete information up-front about every agent's valuation function.

In comparison, an *iterative* GVA can terminate with the same outcome (allocation and payments) but with less information revelation. An iterative auction can elicit information from agents dynamically, as required to determine the efficient allocation. Terminating with the Vickrey outcome provides an iterative procedure with much of the same strategy-proofness as the sealed-bid GVA. The design of an iterative GVA is stated as an important open problem in the auction design literature [BO99, Mil00b]. However, iterative Vickrey auctions are only known for special cases [KC82, DGS86, GS00, Aus00], with restrictions on agent valuation functions. See Table 4.7 for a survey of known results.

Previous attempts to design iterative auctions have (implicitly at least) relied on increasing prices across rounds to compute minimal competitive equilibrium prices, which equal Vickrey payments in special cases. This approach must fail in general problems because there are often *no* single set of minimal CE prices that compute Vickrey payments to every agent. In addition, this approach requires careful price-adjustment, on the "minimal overdemanded set of bundles".

My approach in VickAuction, and the extended *i*Bundle auction, *i*Bundle Extend& Adjust, is to retain the greedy price-updates of *i*Bundle and adjust the prices *after the*

auction terminates to compute minimal competitive equilibrium prices. With this approach the auction does not need to terminate with minimal CE prices. The method also extends to problems without a single set of minimal CE prices that compute Vickrey payments, because we can compute the Vickrey payment to an agent as the minimal price on its bundle over *all* minimal CE prices.

*i*Bundle Extend&Adjust is a simple interpretation of VickAuction, introduced in Chapter 6. The extended *i*Bundle auction collects additional information from agents in order to adjust prices to Vickrey payments. The goal is to implement the Vickrey outcome with best-response agent strategies. The first phase is identical to *i*Bundle(3), and the allocation implemented at the end of the extended auction is that computed at the end of the first phase, which is the efficient allocation when agents follow myopic best-response strategies. The purpose of the second phase is to compute Vickrey payments.

The iterative auction has better information properties than the sealed-bid GVA. In each round agents must only bid for the set of bundles that *maximize* their utility given current ask prices, which does not require agents to compute their exact values for every bundle. Further discussion of agent computation is provided in Chapter 8.

As far as I know, *i*Bundle with Adjust, but without the additional phase, is the first auction with the following property:

Theorem 7.1 (min CE). i*Bundle with* Adjust *computes minimal CE prices and the efficient allocation, for myopic best-response agent strategies as the minimal bid increment* $\epsilon \to 0$.

This follows quite directly from Theorem 6.3 in the previous chapter.

In addition, *i*Bundle Extend&Adjust, which is equivalent to running *i*Bundle with an extended phase and then price-adjust method Adjust*, satisifies the following property:

Theorem 7.2 (Vickrey). i*Bundle Extend&Adjust computes the Vickrey payments and the efficient allocation whenever the agents-are-substitutes condition holds, for myopic best-response agent strategies as the minimal bid increment* $\epsilon \to 0$.

The agents-are-substitutes condition was introduced in the previous chapter (Definition 6.3), and is necessary and sufficient for Vickrey payments to be computed at the minimal

CE prices. Special cases of this theorem capture all known iterative Vickrey auctions for the combinatorial allocation problem (see Table 4.7), including[1]

— linear-additive preferences

— unit-demand preferences

— gross-substitutes preferences

— problems with a single agent in the efficient allocation

— problems with two agents

Theorem 7.2 follows quite directly from the analysis in the previous chapter. The only subtlety here is to prove that the auction will terminate immediately without entering its extended phase when the agents-are-substitutes condition holds. This is proved in Section 7.6.

Finally, I also make the following conjecture for *i*Bundle Extend&Adjust:

CONJECTURE 7.1 (iterative generalized Vickrey auction). i*Bundle Extend&Adjust is an iterative Generalized Vickrey Auction, terminating with Vickrey payments and the efficient allocation for myopic best-response agent strategies as the minimal bid increment* $\epsilon \to 0$.

While a full proof has yet to be completed, experimental results presented in Section 7.7 provide very strong support for the conjecture. I have a proof (see Section 7.6) that *if the extended auction terminates*, then the adjusted prices are Vickrey payments. This follows quite directly from the properties of VICKAUCTION. What is left to prove is that the method for quiescence detection and the introduction of dummy agents in the second phase of the auction is sufficient to push the final phase of the auction to a state in which the Vickrey test (Definition 6.10) holds.

7.1 Overview

The extended auction has two distinct phases. The first PhaseIs used to determine the efficient (value-maximizing) allocation, while the second-PhaseIs used to determine Vickrey

[1] Bikchandani *et al.* [BdVSV01] show that gross-substitutes is sufficient for the agents-are-substitutes condition.

payments. This transition from PhaseI to PhaseII is designed to be hidden from participants. The basic auction rules across both phases are as in *i*Bundle, and prices increase monotonically between PhaseI and PhaseII. The novelties in the auction design are as follows:

- Agents' payments are *adjusted* after the auction terminates, and agents do not pay their final bid prices. This allows the implementation of non-equilibrium solutions, which is important because the GVA outcome cannot always be supported in equilibrium.

- Additional competition is introduced during the second phase of the auction, to make the winning agents continue to bid and reveal more information, enabling final prices to be adjusted to Vickrey payments.

Best-response bids from agents provide information about the complementary-slackness conditions in a primal-dual formulation, and can be used to adjust towards an optimal solution.

PhaseII is designed to force active agents to bid higher prices for bundles received in the optimal allocation. With myopic best-response agent strategies the ask prices to all agents remain valid competitive equilibrium prices during PhaseII. PhaseII terminates precisely when there are no active agents, or at least no active agents still bidding in the auction (which indicates that the ask price to those agents cannot be any higher).

The extended *i*Bundle auction uses *dummy agents* to provide continued competition for agents in the efficient allocation and implement the second phase of VickAuction. The dummy agents are designed to make agents in the efficient allocation continue to bid higher prices until there is enough information to compute Vickrey payments. The method to introduce dummy agents, although experimental at this stage, does seem to succeed in forcing the auction to the Vickrey state and computing Vickrey payments after termination.

This dummy agent method is adopted instead of the straightforward price-update rules in the second phase of VickAuction because it is important that bidders cannot *detect* the transition from PhaseI to PhaseII. An agent's bids in PhaseII decrease the final price paid by other agents, but have no other effect on the outcome of the auction. Thus, if an agent knows it is in PhaseII it might decide to drop out of the auction because of

	A	B	AB
Agent 1	0	a	b
Agent 2	10	0	10
Agent 3	0	0	15

Table 7.1: Problem 8.

participation costs from continued bidding. Another possibility is that an agent might attempt collusive manipulation with another agent. This is discussed below in Section 7.8.1.

The tricky part of the proof of optimality of *i*Bundle Extend&Adjust is to show that the competition from the dummy agents is sufficient to make PhaseII terminate, i.e. to push the bid prices for all active agents high enough to satisfy conditions to compute Vickrey payments. This remains a conjecture.

At the end of the chapter I discuss a number of refinements that may boost computational performance with little loss in incentive and efficiency properties.

7.2 Manipulation of *i*Bundle

Up to this point I have assumed that agents follow myopic best-response strategies, truthfully revealing their demand in response to ask prices in each round of *i*Bundle. The assumption allowed a connection between agent bids, complementary-slackness conditions, and primal-dual optimality.

However, *i*Bundle leaves open the possibility of agent manipulation. *i*Bundle terminates with competitive equilibrium (CE) prices, often minimal CE prices, but this is not always enough to prevent successful manipulation. Alternative strategies available to agents include: placing jump bids, signaling false intentions, or waiting to bid, all of which can reduce economic efficiency and require quite complex game-theoretic reasoning by agents.

Let us consider Problem 8 in Table 7.1, with $a = b = 10$. Suppose that agents 2 and 3 follow a myopic best-response strategy and consider the options available to agent 1. The efficient allocation is $\mathbf{S}^* = (B, A, \emptyset)$, for value $V^* = 20$. Let $(p_1, p_2, p_3) = (p_1(B), p_2(A), p_3(AB))$. In competitive equilibrium, the prices must satisfy: $p_1 \leq 10$, $p_2 \leq 10$, $p_3 \geq 15$, and $p_1 + p_2 \geq p_3$. One set of competitive equilibrium prices are:

$p_1 = 8, p_2 = 8, p_3 = 15$.

Agent 1 might choose to follow myopic best-response. In this case *i*Bundle terminates with one agent paying 7 and the other paying 8, and agent 3 unwilling to pay 16 for bundle AB. Agent 1 can do better by *waiting* while agent 2 bids against agent 3, and then bidding for B to stop agent 3 winning AB when agent 2 has bid 10 for A and can bid no higher. This "slow straightforward" bidding strategy [Mil00a] allows agent 1 to reduce the price that it pays from 7 to 5, while agent 2 pays 10. Agent 1 is said to *free-ride* off the bids of agent 2 and ends up sharing less of the cost of out-bidding the third agent.

The Vickrey payments in this problem are \$5 for each agent, i.e. $p_{\text{vick}}(1) = v_1(B) \Leftrightarrow (V^* \Leftrightarrow (V_{-1})^*) = 10 \Leftrightarrow (20 \Leftrightarrow 15) = 5$, and $p_{\text{vick}}(2) = v_2(A) \Leftrightarrow (V^* \Leftrightarrow (V_{-2})^*) = 10 \Leftrightarrow (20 \Leftrightarrow 15) = 5$. These prices are precisely what agents 1 and 2 might hope to achieve with a slow straightforward bidding strategy if the other agent follows its myopic best-response strategy.

Computing Vickrey payments at the end of the auction with myopic best-response strategies makes myopic best-response becomes a *Bayesian-Nash equilibrium* of the iterative auction. Informally, each agent does as well as it could hope to do with any other strategy, given that the other agents follow myopic best-response strategies.

Milgrom [Mil00a] has earlier observed that in cases in which the minimal CE prices are not unique an agent's optimal strategy is this "slow straightforward" bidding. A slow straightforward strategy submits a bid only when the auction is about to terminate, and the agent is not currently receiving a bundle in the provisional allocation. The cases without unique minimal CE prices are precisely those in which Vickrey payments are not supported in CE.

7.3 *i*Bundle Extend & Adjust: Description

*i*Bundle Extend&Adjust has two distinct phases: PhaseI, in which the final allocation is determined, followed by PhaseII, in which final payments are determined. PhaseI is identical to *i*Bundle(3), the variation of *i*Bundle that maintains separate ask prices for each agent throughout the auction. PhaseI ends when *i*Bundle terminates, at which point the auctioneer stores the provisional allocation. This allocation is implemented at the end of the auction.

The purpose of PhaseII is to collect enough additional information to be able to compute Vickrey payments. At the end of PhaseII, payments to agents are computed as the bid prices at the end of PhaseI minus a discount, which is computed during PhaseII. Both phases follow the price update rules, bidding rules, and winner-determination rules of *i*Bundle. The termination condition in PhaseII, and additional steps performed during each round in Phase II to compute price discounts are new.

Let $\mathbf{S}^* = (S_1^*, \ldots, S_I^*)$ denote the allocation at the end of PhaseI, P^* denote the auctioneer's revenue, $W^* \subseteq \mathcal{I}$ denote the set of agents that receive a bundle in $\mathbf{S}^*$, $(P_{-i})^*$ denote the value of the revenue maximizing allocation without agent i at the current ask prices, and $(W_{-i})^*$ denote the agents in this second-best allocation. Also, let $p^I_{\text{bid},i}(S)$ denote agent i's bid price for bundle S at the end of PhaseI.

As in Chapter 6, I will refer to the *dependents* of agent i as the agents that receive a bundle in allocation $\mathbf{S}^*$ but not in the second-best allocation without agent i at the current ask prices. A *needy* agent is an agent that is in allocation $\mathbf{S}^*$ and has a non-zero adjusted price for its bundle. Finally, an *active* agent is an agent that is still bidding at the current prices.

PhaseI: *i*Bundle(3)

PhaseI is *i*Bundle(3), with termination under the same conditions and unique prices for each agent in every round of the auction. The allocation at the end of PhaseI is stored, and finally implemented at the end of PhaseII.

PhaseII: Extend&Adjust

PhaseII of *i*Bundle Extend&Adjust shares many features with PHASEII of the primal-dual algorithm VICKAUCTION to compute Vickrey payments with best-response agent bids. The purpose of PhaseII is to compute the discount from agent prices at the end of PhaseI to adjust to Vickrey payments.

The final price for agent i at the end of PhaseII is discounted from its final bid price at the end of PhaseI by the *sum* of its *initial discount* $\Delta_{\text{init}}(i)$, computed at the start of PhaseII, and an additional discount $\Delta_{\text{init}}(i)$ computed during PhaseII.

At the start of PhaseII the initial discount, $\Delta_{\text{init}}(i)$, is computed as:

$$\Delta_{\text{init}}(i) = \begin{cases} P^* \Leftrightarrow (P_{-i})^* & \text{, if } i \in W^* \\ 0 & \text{, otherwise} \end{cases}$$

and $\Delta_{\text{extra}}(i) = 0$ for all agents. The dependents for agents $i \in W^*$ are computed as $\alpha(i) = W^* \setminus ((W_{-i})^* \bigcup i)$, with $\alpha(i) = \emptyset$ otherwise. The *needy* agents are those agents in W^* for which $p^I_{\text{bid},i}(S^*_i) \Leftrightarrow (\Delta_{\text{init}}(i) + \Delta_{\text{extra}}(i)) > 0$.

The auctioneer introduces *dummy agents* to drive competition with agents past the end of PhaseI, and push prices into the state where the Vickrey test (Definition 6.10) holds. The auctioneer simulates the dummy agents, generating bids in each round. These bids are not visible to agents.

The auctioneer first introduces a dummy agent for any agent that dropped out of the auction in the last round of PhaseI. Additional dummy agents are introduced dynamically at the end of each round.

A simple rule is used to construct the valuation function of a dummy agent:

DEFINITION 7.1 [dummy agent] The valuation function of a dummy agent constructed to mimic agent j is based on the final ask prices of agent j: set $v(S) = p_{\text{ask},j}(S) + L$ for bundles S with $p_{\text{ask},j}(S) > 0$, and $v(S) = 0$ for all other bundles, for some large constant $L > 0$.

The auctioneer updates the set of *active agents*, and performs the following steps at the end of each round of PhaseII:

1. Compute the new second-best allocation without each *needy* agent in turn, restricting attention to only the *real agents* (ignoring the dummy agents). Update $(W_{-i})^*$, and compute the new dependents of each *needy* agent, comparing the agents in the second-best allocation with the *agents in the efficient allocation.*

2. For each *needy* agents with *active dependents*, increment $\Delta_{\text{extra}}(i)$ by $\sum_{j \in \alpha(i)} \Delta_{\text{incr}}(j)$ where $\Delta_{\text{incr}}(j) \geq 0$ is the increase in bid price by agent j for bundle S^*_j (the bundle it receives in the allocation at the end of PhaseI) since the previous round.

3. Remove agent i from the set of *needy* agents if $p^I_{\text{bid},i}(S^*_i) \Leftrightarrow (\Delta_{\text{init}}(i) + \Delta_{\text{extra}}(i)) \leq 0$.

Test for termination: PhaseII terminates when there no *needy* agents with *active dependents.* Special cases of this termination condition hold where there are: no needy agents;

no active agents; no dependent agents, etc.

Otherwise, the auctioneer introduce *dummy agents* according to the following rules:

(1) for any agent that has just dropped out of the auction, i.e. that was *active* in the previous round but is no longer *active*. Any dummy agent that already existed for this agent is replaced with a new one.

(2) in a state of *quiescence* for the active agents,[2] in which case a dummy agent is introduced for: (*i*) an agent with no dummy that is not active; or failing that (*ii*) an active agent with no dummy; or failing that (*iii*) an active agent that already has at least one dummy agent.

After termination allocation $\mathbf{S}^*$, as computed at the end of PhaseI is implemented, and the final adjusted prices are:

$$p_{\text{adjust}}(i) = \max\left[0,\ p^I_{\text{bid},i}(S^*_i) \Leftrightarrow (\Delta_{\text{init}}(i) + \Delta_{\text{extra}}(i))\right]$$

Worked examples of *i*Bundle Extend&Adjust are provided below.

7.3.1 Discussion

The precise definition of quiescence is not too important. I consider that the auction is in quiescence if: the same active agents have participated in the auction for the past three rounds; and all participating active agents have been allocated the same (non-empty) bundle in the provisional allocation in the past three rounds, and for the same price.

7.3.2 Variation: *i*Bundle(2) and PhaseII

One problem with the extended *i*Bundle auction is that the first PhaseIs *i*Bundle(3), which maintains separate prices for each agent, and can take longer to converge than *i*Bundle(2), which has more direct feedback between agents.

In order to apply PhaseII and the ADJUST* method to *i*Bundle(2), with anonymous prices, we might first build a set of individual ask prices for each agent, $p_i(S)$, for bundle S. One approach is to initialize them to the anonymous prices, and then try to adjust the prices towards prices that are approximately competitive equilibrium and negative-translations of agent valuation functions; for example, reducing the price on bundles than

[2] It is possible that there is a bidding war between the dummy agents and non-active agents without displacing the allocations of active agents.

an agent *does not bid* in the final allocation as far as possible. The goal is to compute individual prices that allow the ADJUST* algorithm to compute Vickrey payments.

Information in bids placed in earlier rounds of the auction can be used to adjust prices. For example, if an agent bids for bundle S_1 at price p_1 in an earlier round, but not for bundle S_2 at price p_2, then this indicates that $v(S_1) \Leftrightarrow p_1 \geq v(S_2) \Leftrightarrow p_2$, and $v(S_1) \Leftrightarrow v(S_2) \geq p_1 \Leftrightarrow p_2$. Now, if the agent bids for bundle S_1 but not S_2 at the final prices, then the final price $p^I(S_2)$ on bundle S_2 can be reduced at least until $p^I(S_2) = p^I(S_1) \Leftrightarrow (p_1 \Leftrightarrow p_2)$.

Similarly, we can reduce prices to an agent not in the final allocation to the prices in the first round in which the agent placed no bids.

7.3.3 Worked Examples

It is useful to demonstrate *i*Bundle Extend&Adjust on Problem 8 in Table 7.1, for different values of a and b. In each case the auction terminates with Vickrey payments for myopic best-response agent strategies.

- Case $(a = b = 3)$. *PhaseI:* $\mathbf{S}^* = (\emptyset, \emptyset, AB)$, $P^* = 13$, $W^* = \{3\}$, $\mathbf{p}^I_{\text{bid}} = (0, 0, 13)$. *PhaseII:* First, compute: $(\mathbf{S}_{-3})^* = (B, A, \emptyset)$, $(P_{-3})^* = 13$, $(W_{-3})^* = \{1, 2\}$, $\alpha(3) = \emptyset$, $\Delta_{\text{init}}(3) = 13 \Leftrightarrow 13 = 0$. Terminates immediately because agent 3 is the only agent in the efficient allocation, and therefore there are no dependents. The outcome is to allocate bundle AB to agent 3 for $p_3 = 13 \Leftrightarrow (0 + 0) = 13$, which is the Vickrey payment, $p_{\text{vick}}(3) = 15 \Leftrightarrow (15 \Leftrightarrow 13) = 13$.

- Case $(a = b = 10)$. *PhaseI:* $\mathbf{S}^* = (B, A, \emptyset)$, $P^* = 15$, $W^* = \{1, 2\}$, $\mathbf{p}^I_{\text{bid}} = (8, 7, 0)$. *PhaseII:* First, compute: $(\mathbf{S}_{-1})^* = (\emptyset, \emptyset, AB)$, $(P_{-1})^* = 15$, $(W_{-1})^* = \{3\}$, $\alpha(1) = \{1, 2\} \setminus \{3, 1\} = \{2\}$, $\Delta_{\text{init}}(1) = 15 \Leftrightarrow 15 = 0$, $(\mathbf{S}_{-2})^* = (\emptyset, \emptyset, AB)$, $(P_{-2})^* = 15$, $(W_{-2})^* = \{3\}$, $\alpha(2) = \{1, 2\} \setminus \{3, 2\} = \{1\}$, $\Delta_{\text{init}}(2) = 15 \Leftrightarrow 15 = 0$. Agents 1 and 2 are *active agents.*

  Do not terminate because agents 1 and 2 are *needy*, and both have the other agent as an active dependent. Instead, introduce a *dummy agent* for agent 3, with values $\mathbf{v}_4 = (0, 0, 15 + L)$ for a large $L > 0$. As prices increase agent 1 drops out first, when $p_1(B) > 10$. At this time $\Delta_{\text{extra}}(2) = 2$ because agent 1's bid has increased by 2 since the end of PhaseI. A dummy agent is introduced for agent 1, with values $\mathbf{v}_5 = (0, 10 + L, 10 + L)$. Finally, agent 2 drops out when $p_2(A) > 10$, at which

time $\Delta_{\text{extra}}(1) = 3$ because agent 2's bid has increased by 3 since the end of PhaseI. PhaseII terminates because there are no active agents.

The outcome is to allocate item B to agent 1 for $p_1 = 8 \Leftrightarrow (0 + 3) = 5$ and item A to agent 2 for $p_2 = 7 \Leftrightarrow (0 + 2) = 5$. These are the Vickrey payments: $p_{\text{vick}}(1) = p_{\text{vick}}(2) = 10 \Leftrightarrow (20 \Leftrightarrow 15) = 5$.

- Case ($a = b = 20$). *PhaseI* is the same as in case $a = b = 10$. *PhaseII* As in case $a = b = 10$, introduce a dummy agent for agent 3, with values $\mathbf{v}_4 = (0, 0, 15 + L)$ for a large $L > 0$. This time, as prices increase agent 2 drops out first, when $p_2(A) > 10$ and $\Delta_{\text{extra}}(1) = 3$. Introduce a dummy agent for agent 2 with value $\mathbf{v}_5 = (10 + L, 0, 10 + L)$. Finally, agent 1 enters $(\mathbf{S}_{-2})^*$, when $p_1(B) = 15$ and $\Delta_{\text{extra}}(2) = 7$. At this stage agent 2 is no longer *needy*, because its total discount $\Delta_{\text{init}}(2) + \Delta_{\text{extra}}(2)$ is equal to its bid price at the end of PhaseI.

  The outcome is to allocate item B to agent 1 for $p_1 = 8 \Leftrightarrow (0 + 3) = 5$ and item A to agent 2 for $p_2 = 7 \Leftrightarrow (0 + 7) = 0$. These are the Vickrey payments: $p_{\text{vick}}(1) = 20 \Leftrightarrow (30 \Leftrightarrow 15) = 5$ and $p_{\text{vick}}(2) = 10 \Leftrightarrow (30 \Leftrightarrow 20) = 0$.

7.4 Iterative Vickrey Auctions

In Chapter 4 I surveyed previous results in the design of iterative Vickrey auctions (see Table 4.7). Iterative Vickrey auctions are known for linear-additive, unit-demand, and gross-substitutes agent preferences. All these auctions assume that agents will follow myopic best-response bidding strategies, and compute Vickrey payments based on those strategies. *i*Bundle Extend & Adjust is an iterative Vickrey auction in all these cases, because the agents-are-substitutes condition introduced in Section 6.1 holds.

It is useful to define the concept of *myopic-implementation* of the Vickrey outcome (the efficient allocation and the Vickrey payments) in an iterative auction:

DEFINITION 7.2 [myopic-implementation] Auction $\mathcal{A}$ myopically-implements the Vickrey outcome if the auction terminates with the Vickrey outcome for agents that follow myopic best-response bidding strategies.

Let $BR(v_i, p)$ denote the best-response bid for agent i with value $v_i(S)$ for bundles $S \subseteq \mathcal{G}$, given prices $p(S)$ on bundles. Best-response can define a *set* of bundles if the agent

is indifferent across a number of bundles. Call $BR(v_i, p)$ a *truthful* best-response bidding strategy. Also, let $BR(\hat{v}_i, p)$ denote an *untruthful* best-response bidding strategy for agent i, for some valuation function $\hat{v}_i \neq v_i$.

One might imagine that an iterative auction that myopically-implements the Vickrey outcome would share the same strong incentive-compatibility properties as the Vickrey-Clarke-Groves mechanisms, i.e. strategy-proofness such that myopic best-response is a dominant strategy for an agent, optimal whatever the strategies of other agents. In fact, manipulation remains possible in such an auction, because agents do not simply have to select a valuation $\hat{v}_i$ and play a best-response $BR(\hat{v}_i, p)$, but have other options available (such as adjusting their valuation, submitting jump bids, etc.)

Gul & Stacchetti [GS00] propose an iterative Vickrey auction that computes Vickrey payments in cases in which they can be computed in the minimal linear-price competitive equilibrium. The authors prove that Vickrey payments make truthful myopic best-response a Bayesian-Nash equilibrium of the auction.

LEMMA 7.1 *Truthful myopic bidding is a sequentially rational best-response to truthful myopic bidding by other agents in an iterative auction with linear-prices that myopically-implements the Vickrey outcome.*

PROOF. The proof follows quite directly from the strategy-proofness of the GVA. Basically, for any other strategy the agent selects a GVA outcome for some non-truthful valuation function, which is less preferable than the GVA outcome for its true valuation function. See Gul & Stacchetti [GS00] for details. ∎

In other words, Gul & Stacchetti show that if every other agent follows a myopic best-response strategy in their auction, *and* if minimal CE prices compute Vickrey payments, then myopic best-response is the optimal strategy for agent i.

Although the connection between Vickrey payments in an iterative combinatorial auction and incentive-compatibility appears to be widely accepted in the literature, I have not found a general proof. Bikchandani & Ostroy [BO00], for example, state:

"...[in] an ascending-price auction [that] finds the smallest market clearing (Walrasian) prices ... buyers get their marginal product and therefore have the incentive to bid truthfully."

for the case that minimal CE (or Walrasian) prices support Vickrey payments (i.e. agents-are-substitutes holds). While this connection between Vickrey payments and incentive-compatibility is also implicit in Ausubel's [Aus97, Aus00] auctions, Ausubel does also provide careful proofs of the incentive properties of his dynamic auctions.

Assuming for the moment that *i*Bundle Extend&Adjust does indeed implement the outcome of the GVA with myopic best-response agent strategies, I prove the Bayes-Nash incentive-compatibility of the auction:

LEMMA 7.2 (incentive-compatibility). *Truthful myopic bidding is a sequentially rational best-response to truthful myopic bidding by other agents in* i*Bundle Extend&Adjust as bid increment* $\epsilon \to 0$*, if the auction myopically-implements the Vickrey outcome.*

PROOF. Suppose agent $i \in \mathcal{I}$ follows a strategy other than truthful myopic best-response, while the other agents follow truthful myopic best-response. Let $p^I(S)$ denote the prices at the end of PhaseI, $p^{II}(S)$ denote the prices at the end of Phase II, but before prices are adjusted, $p_{\text{adjust},i}(S)$ denote the adjusted prices at the end of PhaseII, and $\hat{\mathbf{S}} = (\hat{S}_1, \ldots, \hat{S}_I)$ denote the allocation computed at the end of PhaseI.

The first step in the proof is to construct a valuation function $\hat{v}_i$ for agent i, for which $(p_{\text{adjust},i}(\hat{S}_i), p^{II}_{-i}(\hat{S}_{-i}))$ are in competitive equilibrium with allocation $\hat{\mathbf{S}}$ given agent preferences $(\hat{v}_i, v_{-i})$, i.e. agent i with value $\hat{v}_i$ and agents $j \neq i$ with values v_j.

Consider the valuation function $\hat{v}_i$ defined by:

$$\hat{v}_i(S) = \begin{cases} p^{II}_i(\hat{S}_i) & \text{, if } S = \hat{S}_i \\ p^{II}_i(\hat{S}_i) & \text{, if } S \supset \hat{S}_i \\ 0 & \text{, otherwise} \end{cases}$$

Prices $(p^I_i(\hat{S}_i), p^{II}_{-i}(\hat{S}_{-i}))$ form a competitive equilibrium with allocation $\hat{\mathbf{S}}$ and agent preferences $(\hat{v}_i, v_{-i})$. (CS1) holds for agent i with preferences $\hat{v}_i$ because $p^I_i(\hat{S}_i) \leq p^{II}_i(\hat{S}_i) = \hat{v}_i(\hat{S}_i)$, and $p^{II}_i(S') \geq p^I(\hat{S}_i), \quad \forall S' \supseteq \hat{S}_i$. (CS1) holds for agents $j \neq i$ at the end of PhaseI, and at the end of PhaseII because the agents continue to follow myopic best-response strategies and prices p^{II}_j are negative translations of agent's valuation functions. (CS2) holds because allocation $\hat{\mathbf{S}}$ maximized revenue to the auctioneer at the end of PhaseI, and continues to maximize revenue at prices (p^I_i, p^{II}_{-i}) because the price $p^{II}_j(S)$ on all bundles $S \neq \hat{S}_j$ increases by less than the price on bundle $\hat{S}_j$ during PhaseII.

By the analysis of the ADJUST procedure in the previous chapter (see Section 6.2.1), prices $(p_i^I(\hat{S}_i) \Leftrightarrow (P^* \Leftrightarrow (P_{-i})^*), p_{-i}^{II}(\hat{S}_{-i}))$ are also in CE with allocation $\hat{\mathbf{S}}$ for agent preferences $(\hat{v}_i, v_{-i})$; where P^* is the revenue from allocation $\hat{\mathbf{S}}$ at prices $(p_i^I(\hat{S}_i), p_{-i}^{II}(\hat{S}_{-i}))$, and $(P_{-i})^*$ is the value of the revenue-maximizing allocation without agent i at prices, p_{-i}^{II}, at the end of PhaseII

The adjusted price $p_{\text{adjust},i}(\hat{S}_i)$ at the end of PhaseII is equal to $p_i^I(\hat{S}_i) \Leftrightarrow (P^* \Leftrightarrow (P_{-i})^*$ because it is computed as $p_i^{II}(\hat{S}_i) \Leftrightarrow \Delta(i)$, where $\Delta(i) = P^* + \delta \Leftrightarrow (P_{-i})^*$, for $\delta = p_i^{II}(\hat{S}_i) \Leftrightarrow p_i^I(\hat{S}_i)$. Thus, $(p_{\text{adjust},i}(\hat{S}_i), p_{-i}^{II}(\hat{S}_{-i}))$ are in CE with allocation $\hat{\mathbf{S}}$ for agent preferences $(\hat{v}_i, v_{-i})$.

The second-step in the proof is to show that agent i's utility with truth-revelation in the GVA is greater than its utility at the outcome of the auction, i.e. $v_i(\hat{S}_i) \Leftrightarrow p_{\text{adjust},i}(\hat{S}_i)$. First consider its GVA payment with a report of $\hat{v}_i$, when the other agents report truthful values v_{-i}. The Vickrey outcome in this case is allocation $\hat{\mathbf{S}}$, as computed in the auction, and agent i's utility is

$$\begin{aligned} u_i(\hat{v}_i) &= v_i(\hat{S}_i) \Leftrightarrow p_{\text{vick},i}(\hat{v}_i, v_{-i}) \\ &\geq v_i(\hat{S}_i) \Leftrightarrow p_{\text{adjust},i}(\hat{S}_i) \end{aligned}$$

The inequality follows because the Vickrey payment is smaller than the minimal price over all minimal CE prices.

Finally, it follows from the strategy-proofness of the GVA that

$$u_i(v_i) \geq u_i(\hat{v}_i), \quad \text{for all } \hat{v}_i \neq v_i$$

and therefore agent i's utility from truth-revelation in the GVA is greater than its utility from outcome $(p_{\text{adjust},i}, \hat{S}_i)$ in the auction. This establishes that for agent i truthful myopic bidding is a sequentially-rational best-response in equilibrium with truthful myopic bidding from other agents, under the assumption that the auction myopically-implements the Vickrey outcome. ▌

In other words, terminating in Vickrey payments provides quite a high degree of incentive-compatibility, but *not* full strategy-proofness. A method is introduced in the next section to restrict agent strategies and make a slightly stronger claim about the robustness-to-manipulation of an iterative Generalized Vickrey Auction.

7.5 Proxy Agents: Boosting Strategy-Proofness

Moving from single-shot Vickrey mechanisms to iterative Vickrey mechanisms makes it necessary to accept a loss in *full* strategy-proofness. Full strategy-proofness requires that all agents simultaneously *commit* to a (possibly untruthful) valuation function, which conflicts with the desire to allow agents to reveal incremental information.

Ideally, we would like to restrict agents to follow a (possibly untruthful) best-response strategy, for some *ex ante* fixed valuation function. Such a restriction, if possible, would allow the following strong claim about strategy-proofness:

LEMMA 7.3 *Truthful myopic bidding is a* dominant strategy *in an iterative Vickrey outcome if agents are restricted to follow a myopic best-response strategy for some* ex ante *fixed (but perhaps untruthful) valuation function* $\hat{v}(\cdot)$.

One sure way to enforce this restriction is to introduce a proxy-bidding agent interface into the auction, which requires a bidding agent to provide a complete valuation function up-front, and then follows a myopic best-response strategy with that value information in the auction. However, this would transform the iterative auction into a single-shot mechanism, and lose the incremental information revelation properties.

A middle ground, which provides some additional strategy-proofness over-and-above Bayesian-Nash incentive-compatibility, but without providing complete strategy-proofness, is to restrict an agent to follow a myopic best-response strategy that is at least *consistent* with a single valuation function across all rounds.

One might imagine two ways to restrict agent i to a best-response strategy for some consistent valuation function $\hat{v}_i(\cdot)$.

- Introduce additional bidding rules, for example preventing "jump bids" by making an agent bid at the current ask price; and check that an agent's bids across multiple rounds in the auction as prices change are consistent with a best-response strategy for a particular valuation function.

- Provide semi-autonomous proxy bidding agents, one for each agent, that receive incremental value information from agents and follow a myopic best-response strategy consistent with that information.

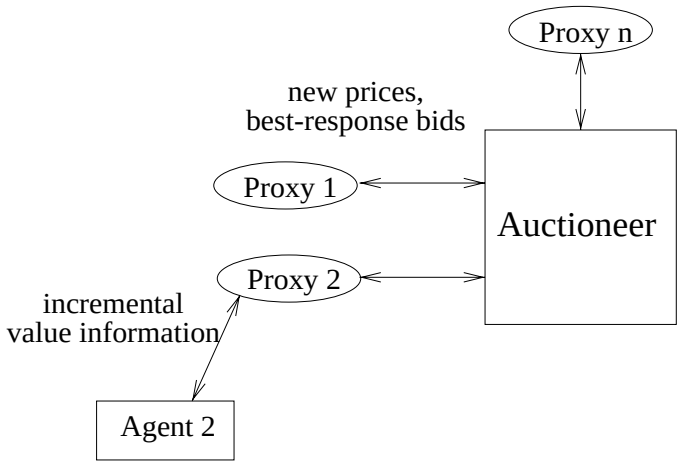


Figure 7.1: Proxy bidding agents.

In Parkes & Ungar [PU00b] we pursued the idea of semi-autonomous proxy bidding agents, that sit between agents and the auctioneer, and submit best-response bids whenever they have enough information about an agent's (possibly untruthful) valuation function to determine the utility-maximizing bundle(s) at the current prices (Figure 7.1). Essentially the proxy agents transform the iterative auction into an *iterative direct-revelation* mechanism, in which agents report incremental information about their values for different bundles. In comparison, the classic mechanism design literature has typically considered only *single-shot* direct-revelation mechanisms.

Semi-autonomous proxy agents retain the computational advantages of iterative auctions because agents can provide incremental information about value; looking ahead to the next chapter, an iterative auction with proxy bidding agents remains *bounded-rational compatible*— an agent can follow its optimal strategy with an approximate valuation function.

The following result is a direct consequence of Lemma 7.2, the Bayesian-Nash incentive-compatibility of an iterative combinatorial auction that myopically-implements the Vickrey outcome:

THEOREM 7.3 *Truthful dynamic information revelation is a sequentially rational best-response to truthful dynamic information revelation by other agents in an iterative auction $\mathcal{A}$ with best-response proxy bidding agents that myopically-implements the GVA.*

The restriction to best-response strategies does not itself strengthen the incentive-compatibility properties of an iterative Vickrey auction. In one extreme (and unachievable) case, if the proxy agents are able to force agents to provide incremental value information consistent with a single *ex ante* fixed valuation function then we can make the following claim:

PROPOSITION 7.1 (dominant strategy). *Truthful dynamic information revelation is a dominant strategy in an iterative auction $\mathcal{A}$ with best-response proxy bidding agents that myopically-implements the GVA, when agents must provide information consistent with an* ex ante *fixed (but perhaps untruthful) valuation function.*

In other words, incremental truth-revelation is a dominant strategy so long as the decisions made by other agents about how to misrepresent their values for bundles are not conditioned on observed information during the auction. This is a stronger claim than Bayesian-Nash (Lemma 7.2), which states that myopic best-response is sequentially-rational in equilibrium with myopic best-response from other agents, but weaker than the full strategy-proofness of the GVA.

One might imagine a method in which an agent is made to commit to a particular "manipulation function", a particular mapping from values to reported values, before it computes its actual values for different bundles. This manipulation function could also reside in the proxy agent. However, this manipulation function would only provide the required property of an *ex ante* fixed valuation if used in combination with a method to validate that incremental information provided by an agent to its proxy was *truthful* information, which flies in the spirit of mechanism design.

A middle ground can be achieved with proxy bidding agents that:

(a) enforce *self-consistency* in information reported across rounds

(b) require an agent to provide enough value information in each round to enable the proxy agent to determine a bundle(s) that maximizes utility given prices in the current round, for all possible valuations consistent with the current approximate information.

Here is a reasonable proposition about the incentive properties of such a proxied iterative Vickrey auction:

PROPOSITION 7.2 *Given auction $\mathcal{A}$, that myopically-implements the Vickrey outcome,*

introducing proxy bidding agents with consistency checks "limits" the opportunities for successful manipulation.

Intuitively, in every round that an agent reports more value information it commits itself to a smaller set of possible reported valuation functions, and restricts its ability to condition future announcements on information revealed by other agents. Providing a theoretical and/or empirical measure of "limits" in Proposition 7.2 is left for future work.

7.5.1 Consistency Checking and Best-response

Formally, let us consider what is required for a proxy agent to: (a) have enough information to compute a best-response bid; and (b) check information consistency across rounds.

Let $\hat{v}^1_{\text{approx},i}, \hat{v}^2_{\text{approx},i}, \ldots$, denote the sequence of approximate valuation information provided by agent i, in rounds 1, 2, etc.

Given an approximate valuation function $\hat{v}_{\text{approx},i}$, let $\mathcal{C}(\hat{v}_{\text{approx}}) \subseteq \mathcal{V}$ denote the set of completely specified valuation functions that are *compatible* with approximate information $\hat{v}_{\text{approx}}$, where $\mathcal{V}$ is the set of all possible valuation functions. The particular definition of *compatible* is that which is natural given the type of approximation, for example if the approximation states upper- and lower- bounds on values, then a compatible value is any value between the bounds.

An approximation is *consistent* with an earlier approximation if all compatible valuations were also compatible in the arlier approximation:

Definition 7.3 [consistent] Approximation v''_{approx} is *consistent* with approximation v'_{approx}, written $v''_{\text{approx}} \subseteq v'_{\text{approx}}$, if the set of compatible values $\mathcal{C}(v''_{\text{approx}}) \subseteq \mathcal{C}(v'_{\text{approx}})$, i.e. if v''_{approx} places a stronger condition on the compatible valuation functions.

The *consistency check* by the proxy agent across rounds is defined as follows:

Definition 7.4 [consistency check] For consistency from round t to round $t+1$, the proxy agent requires

$$\hat{v}^{t+1}_{\text{approx},i} \subseteq \hat{v}^t_{\text{approx},i}$$

such that the approximation in round $t+1$ is consistent with the approximation in round t.

In words, the information in round $t+1$ must be a refinement of the information in round t, and therefore consistent with the information in all previous rounds by transitivity. Each new piece of information must remove valuations from the reachable set without introducing new possibilities.

Given prices $p_i^t(S)$ to agent i for bundles $S \subseteq \mathcal{G}$ in round t, the auctioneer requires enough information from agent i to compute a best-response that is optimal for *all future refinements*.

DEFINITION 7.5 [best-response information requirement] In round t agent i must provide enough information, $\hat{v}^t_{\text{approx},i}$, to allow a single bundle to solve the best-response problem, for *all* valuations consistent with the approximation and for the current prices.

In other words, the agent's best-response must be the same for all valuation functions consistent with its current approximate value information at the current prices.

Discussion

I have provided definitions for a *worse-case* framework; i.e. a new approximation is only consistent with an old approximation if there are *no* new compatible valuations— not even one. Similarly, the best-response condition states that there must be a single best-response for *every* future set of consistent approximations.

These definitions are not suitable with some probabilistic approximations, such as "the value for bundle S is Normally distributed with mean μ and standard deviation σ". More suitable definitions would replace the *worst-case* guarantees with "with high probability" guarantees. For example, a new approximation might be said to be δ-consistent with a current approximation if the probability that a valuation consistent with the new approximation was also consistent with the previous approximation is *at least* $1 \Leftrightarrow \delta$. Consistency might also require a probabilistic approximation with *less variance* and a consistent mean, such that the distributions converge to a single consistent point in valuation space.

7.5.2 Special Case: Upper and Lower Bounds

An important special case occurs when an agent can provide approximate information in the form of *upper-* and *lower-* bounds on value.

Bounds $[\underline{v}(S), \overline{v}(S)]^t$ in round t denote lower bounds $\underline{v}(S) \leq \hat{v}(S)$ on bundles S and upper bounds $\overline{v}(S) \geq \hat{v}(S)$, for some (perhaps untruthful) valuation function $\hat{v}(S)$, and

every bundle $S \subseteq \mathcal{G}$. Valuation function $\hat{v}(S)$ is *compatible* with bounds if $\hat{v}(S) \geq \underline{v}(S)$ and $\hat{v}(S) \leq \overline{v}(S)$ for all $S \subseteq \mathcal{G}$.

Given prices $p_i(S)$ and bounds $[\underline{v}_i(S), \overline{v}_i(S)]$), let $\underline{u}_i(S) = \underline{v}_i(S) \Leftrightarrow p_i(S)$ and $\overline{u}_i(S) = \overline{v}_i(S) \Leftrightarrow p_i(S)$, denote the lower and upper bounds on utility.

In order to formulate the rules for sufficient information to compute a best-response bid that is optimal for *all* consistent valuations with bounds on value, it is useful to define a bundle with *strict-positive* value:

DEFINITION 7.6 [strict-positive value] Bundle S has strict-positive value $v_i(S)$ if the value on the bundle is (strictly) greater than the value for all bundles contained in S, i.e. if $v_i(S) > v_i(S')$ for all $S' \subset S$.

Intuitively, the bundles with strict-positive value are those bundles that are *important* for an agent to consider when constructing its best-response bid set. Let $\mathcal{SP}$ denote the set of bundles with strict-positive value to an agent. Given this, then value bounds provide sufficient information to compute a best-response bid, if the following conditions hold on the *utility* bounds of every strict-positive bundle $S \in \mathcal{SP}$:

$$\begin{array}{lllr} \text{either} & \forall T \neq S,\ T \in \mathcal{SP}, & \underline{u}_i(S) + \epsilon \geq \max(0, \overline{u}_i(T)) & \text{(dominates)} \\ \text{or} & \exists\ T \neq S, T \in \mathcal{SP}, & \overline{u}_i(S) \leq \max(\epsilon, \underline{u}_i(T) + \epsilon) & \text{(is dominated)} \end{array}$$

In words, this states that every bundle must either:

— have a utility that dominates the utility of all other bundles, for all future refinements, *and* is positive

— or, have a utility that is either dominated by at least one other bundle for all future refinements *or* negative.

As special-cases: if a bundle's upper-bound on utility is no greater than ϵ the proxy agent can know not to bid for that bundle; and if a bundle's lower-bound on utility is negative (in fact less than $\Leftrightarrow\epsilon$) then the agent cannot bid for that bundle without refinining its value.

Clearly, the best-response bid when these conditions do hold on every strict-positive valued bundle is to *bid for the bundles that satisfy the (dominates) condition.*

	A	B	AB
Agent 1	0	14^*	14
Agent 2	10^*	0	10
Agent 3	4	5	12

Table 7.2: Problem 9.

7.5.3 Example: Incremental Information Revelation

This section presents a worked example of *i*Bundle(2) with proxy bidding agents on Problem 9 in Table 7.2, in which the efficient allocation $\mathbf{S}^* = (A, B, \emptyset)$.

Assume that the agents initially provide the following information to their proxy agents:

	A	B	AB
agent 1	0	$[13.5, 14.5]$	same as B
agent 2	$[2, 12]$	0	same as A
agent 3	$[2, 6]$	$[2, 6]$	$[8, 16]$

Assume that the minimal bid increment, $\epsilon = 2$. The proxied auction proceeds automatically through 7 rounds with this information, as illustrated below:

Round	Prices			Bids			Selected utility bounds		
	A	B	AB	Agent 1	Agent 2	Agent 3	Agent 1	Agent 2	Agent 3
1	0	0	0	$(B, 0)^*$	$(A, 0)^*$	$(AB, 0)$	B:[13.5, 14.5]	A:[2, 12]	A:[2, 6] AB:[8, 16]
2	0	0	2	$(B, 0)$	$(A, 0)$	$(AB, 2)^*$	B:[13.5, 14.5]	A:[2, 12]	A:[2, 6] AB:[6, 14]
3	2	2	2	$(B, 2)^*$	$(A, 2)^*$	$(AB, 2)$	B:[11.5, 12.5]	A:[0, 10]	A:[0, 4] AB:[6, 14]
4	2	2	4	$(B, 2)^*$	$(A, 2)^*$	$(AB, 4)$	B:[11.5, 12.5]	A:[0, 10]	A:[0, 4] AB:[4, 12]
5	2	2	6	$(B, 2)$	$(A, 2)$	$(AB, 6)^*$	B:[11.5, 12.5]	A:[0, 10]	A:[0, 4] AB:[2, 10]
6	4	4	6	$(B, 4)^*$	$(A, 4)^*$	$(AB, 6)$	B:[9.5, 10.5]	A:[-2, 8]	A:[-2, 2] AB:[2, 10]
7	4	4	8	$(B, 4)^*$	$(A, 4)^*$	$(AB, 8)$	B:[9.5, 10.5]	A:[-2, 8]	A:[-2, 2] AB:[0, 8]
8	4	4	10	$(B, 4)$	$(A, 4)$	?	B:[9.5, 10.5]	A:[-2, 8]	**A:[-2, 2] AB:[-2, 6]**

In rounds 1–7 the proxy agents have enough information to submit a best-response bid (to within ϵ). However, in round 8, the approximate information provided by agent 3 is not sufficient to determine the best-response. Notice that the utility bounds on bundle AB do not satisfy the *(dominates)* condition with respect to the utility bounds on item A (or on item B).

In this round agent 3 must provide more value information to its proxy agent. Only information *consistent* with $\hat{v}_{\text{approx},3}(A) = [2, 6], \hat{v}_{\text{approx},3}(B) = [2, 6], \hat{v}_{\text{approx},3}(AB) = [8, 16]$ is allowed. Suppose that agent 3 provides bounds [11, 16] on bundle AB. This refines the utility bound to [1, 6].

The auction can now proceed as follows:

	A	B	AB	Agent 1	Agent 2	Agent 3	Agent 1	Agent 2	Agent 3
8	4	4	10	$(B, 4)$	$(A, 4)$	$(AB, 10)^*$	B:[9.5, 10.5]	A:[-2, 8]	A:[-2, 2] AB:[1, 6]
9	6	6	10	$(B, 6)$	?	$(AB, 10)$	B:[7.5, 8.5]	**A:[-4, 6]**	A:[-4, 0] AB:[1, 6]

At round 9 more information is required from agent 2. Suppose agent 2 first provides new bounds $v_{\text{approx},2}(A) = [2, 10]$. These bounds are consistent, but not sufficient for best-response because the lower utility bound is still more than ϵ below 0. With information $v_{\text{approx},2}(A) = [6, 10]$ the auction can continue.

	A	B	AB	Agent 1	Agent 2	Agent 3	Agent 1	Agent 2	Agent 3
9	6	6	10	$(B, 6)^*$	$(A, 6)^*$	$(AB, 10)$	B:[7.5, 8.5]	A:[0, 6]	A:[-4, 0] AB:[1, 6]
10	6	6	12	$(B, 6)^*$	$(A, 6)^*$	$(AB, 12)$	B:[7.5, 8.5]	A:[0, 6]	A:[-4, 0] AB:[-1, 4]
11	6	6	14	$(B, 6)$	$(A, 6)$	?	B:[7.5, 8.5]	A:[0, 6]	**A:[-4, 0]** **AB:[-3, 2]**

In round 11 more information is required from agent 3. Suppose that agent 3 provides $v_{\text{approx},3}(AB) = [11, 13]$, which will adjust the utility bounds on AB to [-3, -1]. This is enough information for the agent's proxy agent to compute an *empty* best-response:

	A	B	AB	Agent 1	Agent 2	Agent 3	Agent 1	Agent 2	Agent 3
12	6	6	14	$(B, 6)$	$(A, 6)$	$\emptyset$	B:[7.5, 8.5]	A:[0, 6]	A:[-4, 0] AB:[-3, -1]

and the auction terminates with final allocation $\mathbf{S}^* = (B, A, \emptyset)$, which is the efficient allocation. The final information revealed to the proxy agents in this example is tabulated below.

	A	B	AB
agent 1	0	$[13.5, 14.5]$	same as B
agent 2	$[6, 10]$	0	same as A
agent 3	$[2, 6]$	$[2, 6]$	$[11, 13]$

Notice two interesting effects of the proxy agents:

- The auction has a "multi-modal" interface. An agent can either submit quite accurate information up-front, as is the case for Agent 1 in this example, or provide incremental information as required, as is the case for Agents 2 and 3.
- The auction is now "staged", with a number of rounds performed automatically via communication with the proxy agents but without communication with the actual agents; e.g. round 1–8, 9–11.

7.5.4 Example: Strategic Revelation in a Proxied English Auction

In this section I illustrate the incentive properties of a proxied Vickrey auction with a simple single item example.

Consider a proxy bidding-agent interface into the English auction, which is an ascending-price auction for a single item in which the item is sold to the highest bidder for its bid price. The English auction myopically-implements the Vickrey outcome as the bid increment $\epsilon \to 0$, with the item sold to the highest bidder for ϵ above the second-highest value over all agents.

Introducing proxy agents that accept refinements on upper- and lower- bounds on value from agents makes the auction a staged Vickrey auction. As discussed above:

— iterative truth revelation is a dominant strategy in response to strategies from other agents consistent with *ex ante* fixed valuation functions

— iterative truth revelation is a Bayesian-Nash equilibrium of the proxied auction, i.e. a sequentially-rational best-response to iterative truth revelation from other agents

However, iterative truth-revelation is *not* a dominant-strategy equilibrium. I construct an example below in which an agent can increase its utility with a non-truthful strategy.

Example

Consider an example with 3 agents, with values $v_1 = 5, v_2 = 10$ and $v_3 = 15$. In this simple example it is possible to construct an example of strategies for agents 1 and 2 for which agent 3's best-response is not incremental truth-revelation.

I first consider agent 3's strategy when agents 1 and 2 reveal incremental information consistent with a single fixed (but perhaps untruthful) valuation function. First, suppose that each agent plays the Bayesian-Nash equilibrium, maintaining bounds compatible with its true value. Consider initial bounds [2, 6], [5, 15], [8, 20]. Proxy agents 1, 2 and 3 bid while $p \leq 2$, then agents 2 and 3 bid while $p \leq 5$. Finally, with the price at $p = 5 + \epsilon$, agents 1 and 2 need to provide more information. Suppose agent 1 updates its bounds $[2, 6] \rightarrow [2, 5]$, and agent 2 $[5, 15] \rightarrow [9, 12]$. Proxy agent 1 drops out, while proxy agents 2 and 3 bid while $p \leq 8$. With the price at $p = 8 + \epsilon$, agent 3 provides more information. Suppose agent 3 updates its bounds $[8, 20] \rightarrow [12, 18]$. With this information the price increases to $p = 9 + \epsilon$, and agent 2 might continue refining its lower bound until it is approximately 10. At this point agent 3 wins the auction, and pays $10 + \epsilon$.

Even if agent 2 provides untruthful information, for example consistent with $\hat{v}_2 = 9$ instead of $v_2 = 10$, agent 3's optimal strategy is still incremental truth-revelation. This strategy will win the item for a price of $9 + \epsilon$, which is the best possible outcome for agent 3.

However, the following example demonstrates that truthful incremental information revelation is not a dominant strategy in a proxied iterative Vickrey auction. Suppose that agent 1 follows a truthful strategy, while agent 2's (irrational) strategy is:

set initial bounds $[4, 30]$. *At first request for more information, provide bounds* $[6, 30]$. *If the agent is provisionally allocated the item in the next state of the auction provide bounds* $[25, 30]$, *otherwise provide bounds* $[7, 7]$.

Notice that agent 2 makes a dynamic decision about whether to announce information consistent with a value of 7, or with a value somewhere between 25 and 30.

First, consider the outcome if agent 3 follows the following incremental truthful strategy. Set initial bounds to $[5, 20]$. The price will increase to $4 + \epsilon$, at which point agent 3 is provisionally allocated the item. Agent 2 provides new bounds $[6, 30]$, and the price increases to $5 + \epsilon$, with the item allocated to agent 2. Finally, agent 2 updates its bounds to $[25, 30]$, and the auction will terminate with agent 2 buying the item for price $15 + \epsilon$.

Here is a non-truthful alternative strategy that produces a better outcome for agent 3. Set initial bounds to $[30, 40]$. The price will increase to $4 + \epsilon$, at which point agent 3 is provisionally allocated the item. Agent 2 provides new bounds $[6, 30]$, and the price increases to $6 + \epsilon$, with the item allocated to agent 3. Finally, agent 2 updates its bounds to $[7, 7]$, and the auction will terminate with agent 3 buying the item for price $7 + \epsilon$.

7.5.5 Real World Proxy Agents

On-line auctions such as eBay, www.ebay.com, for consumer-to-consumer e-commerce present a real-world example of auctions with separate valuation and bidding problems: people value items, and eBay provides automated bidding agents that monitor auctions and place bids. In an ascending-price auction, the proxy agents are configured with a user's *reservation value*, the maximum she will pay for an item, and bid while the price is below that value. Interestingly, the proxy agents do not convert ascending-price auctions into sealed-bid auctions because they can inform a user by e-mail when her reservation value has been reached, and accept *updated* values. This allows the user to deliberate further about her value for the item, but only if that is required by the current price in the auction, and makes the auction with proxy agents bounded-rational compatible.

Proxy bidding agents that restrict agents to placing a bid at the current ask price may also be useful in preventing "code bidding" between agents in an auction, to achieve collusive outcomes. This practice of adding "magic numbers" on the trailing digits of bids to pass information to other bidders was observed in the FCC spectrum auction, an open and simultaneous auction for individual licenses [CS00]. The FCC introduced "click-box" bidding to constrain bids to be one of a finite number of bid increments above the current ask price.

7.6 Theoretical Analysis

In this section I provide a proof of Theorem 7.2, that *i*Bundle Extend&Adjust computes the Vickrey outcome whenever the agents-are-substitutes condition holds and the minimal CE prices compute the Vickrey payments.

First, I prove that the auction computes the Vickrey outcome *whenever it terminates*,

irrespective of the agents-are-substitutes condition. This makes significant progress towards establishing Conjectre 7.1, because all that remains to prove (or disprove) is that the current rules for introducing dummy agents is sufficient to force the auction into a condition in which the Vickrey test conditions hold and it terminates, adjusting prices to Vickrey payments.

Lemma 7.4 states that *i*Bundle Extend&Adjust computes the GVA outcome whenever it terminates.

LEMMA 7.4 (optimality if terminates). i*Bundle Extend&Adjust computes the efficient allocation and Vickrey payments for myopic best-response bidding strategies as the minimal bid increment* $\epsilon \to 0$*, whenever the auction terminates.*

PROOF. The final prices in the auction satisfy the *Vickrey test* conditions (Definition 6.10) by construction. The discounts computed during PhaseII of the extended auction are simply equal to the discounts that would be computed with ADJUST* at the end of PhaseII. Therefore, the final adjusted prices are Vickrey payments by Theorem 6.6 whenever the auction terminates. The allocation is the one from the end of PhaseI, as computed with the regular *i*Bundle auction, and therefore efficient by Theorem 5.1. ∎

Lemma 7.5 states that PhaseII will terminate immediately, without any dummy agents, when the agents-are-substitutes condition holds.

LEMMA 7.5 (terminate if agents-are-substitutes). i*Bundle Extend&Adjust terminates whenever the agents-are-substitutes condition holds, for myopic best-response bidding strategies as the minimal bid increment* $\epsilon \to 0$*.*

PROOF. (sketch) The agents-are-substitutes condition implies that there is a *unique* set of minimal CE prices (Lemma 6.5). The initial discounts computed in PhaseII with ADJUST* compute this set of minimal CE prices. There can be no dependent agents and PhaseII will terminate immediately because there can only be dependent agents at minimal CE prices when there are multiple sets of minimal CE prices. ∎

Theorem 7.2, that *i*Bundle Extend&Adjust computes the Vickrey outcome whenever the agents-are-substitutes condition holds, follows immediately from Lemma 7.4 and

Lemma 7.5.

7.7 Experimental Analysis

In this section I describe the results of experiments run with *i*Bundle Extend&Adjust and agents with myopic best-response strategies, on the same problem sets used to test *i*Bundle in Chapter 5. The first set of results measure the distance between minimal CE prices and the result of using ADJUST* on prices at the end of *i*Bundle(3). The results demonstrate that *i*Bundle with adjusted prices computed at the end of the auction terminates with minimal CE prices. The second set of results compare the adjusted prices at the end of the extended auction, i.e. after PhaseII, with prices in the Generalized Vickrey auction. Again, the results show very strong support for Conjecture 7.1, that the extended auction is an iterative Generalized Vickrey auction.

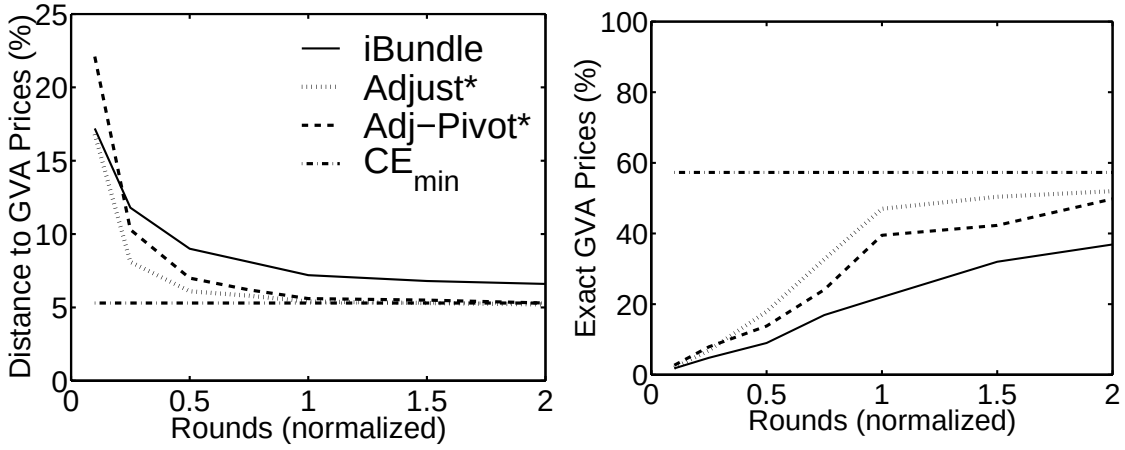
7.7.1 Results I: *i*Bundle and ADJUST*

This section presents the results of experiments to compare the effect of computing adjusted prices at the end of *i*Bundle with computing minimal CE prices and Vickrey payments. The variations ADJUST, ADJUST*, and ADJ-PIVOT*, introduced in the previous chapter, are all considered. In these first set of experiments, first reported in Parkes & Ungar [PU00b], the auction is *not* extended into PhaseII, and only computes Vickrey payments when they are supported in competitive equilibrium.

The auction is tested on problems PS 1–12 in Table 5.5 (Chapter 5) and also problems Decay, Weighted-random (WR), Random and Uniform in Sandholm [San99]. Each problem set defines a distribution over agents' values for bundles of items.

In this set of experiments the distance $\mathcal{D}(p_i(S_i^*), p_{\text{vick}}(i))$ between prices $p_i(S_i^*)$ and Vickrey payments is measured with an L_1 norm, as $L_1(p_i, p_{\text{vick}}) = \sum_i |p_i(S_i) \Leftrightarrow p_{\text{vick}}(i)| / \sum_i v_i(S_i)$, i.e. the sum absolute difference between the price charged to each agent and its GVA price normalized by the total value of the allocation over all agents. An L_1 norm is appropriate because minimal CE prices is computed with a linear additive measure over the auctioneer's price to each agent in the allocation, and because errors in the prices are always one-sided (i.e. greater than the Vickrey payments).

There does not appear to be a useful measure of the distance to Vickrey payments



(a) Distance.

(b) Fraction Correct.

Figure 7.2: Average performance of *i*Bundle with price-adjustment ADJUST* and ADJ-PIVOT* in problems PS 1–12 (see Table 5.5). The number of rounds to termination is varied by adjusting the minimal bid increment.

in problems in which the auction's allocation is inefficient, and different from that in the GVA. Thus, I compute the average distance over problem instances in which *i*Bundle computes the optimal allocation, which approaches 100% of problems as the bid increment gets small.

Figure 7.2 plots the distance to the Vickrey payments in *i*Bundle, before and after price-adjustment using ADJUST* and ADJ-PIVOT*, averaged over 25 trials each of problems PS 1–12. I ran *i*Bundle with different bid increments to vary the number of rounds to termination, and averaged performance across problem sets by normalizing the number of rounds to termination according to the minimal number of rounds in which *i*Bundle achieves 100% allocative efficiency. For comparison, I also plot the distance between minimal CE prices and Vickrey payments.

The results show clear support for Theorem 7.1, which states that *i*Bundle(3) followed by ADJUST* computes Vickrey payments whenever they are supported in minimal CE prices. The average distance between minimal CE prices and GVA prices across these problems is 5.3%. For small bid increments *i*Bundle computes prices to within 6.5% of the Vickrey payments, with ADJUST to within 5.5% (not plotted), and with ADJUST* and ADJ-PIVOT* to within 5.2%. The prices continue to adjust towards the min CE prices for bid increments smaller than those required for 100% allocative efficiency, corresponding to normalized rounds to termination > 1. It is noteworthy that the approximate method

ADJ-PIVOT* is as effective as ADJUST* for small bid increments.

The benefit of ADJUST* over ADJUST is quite marginal without the extended phase of the auction. In comparison, the results with ADJUST* after the second phase of the auction demonstrate the importance of computing price adjustments beyond a single minimal CE price vector (see Figure 7.5 for example).

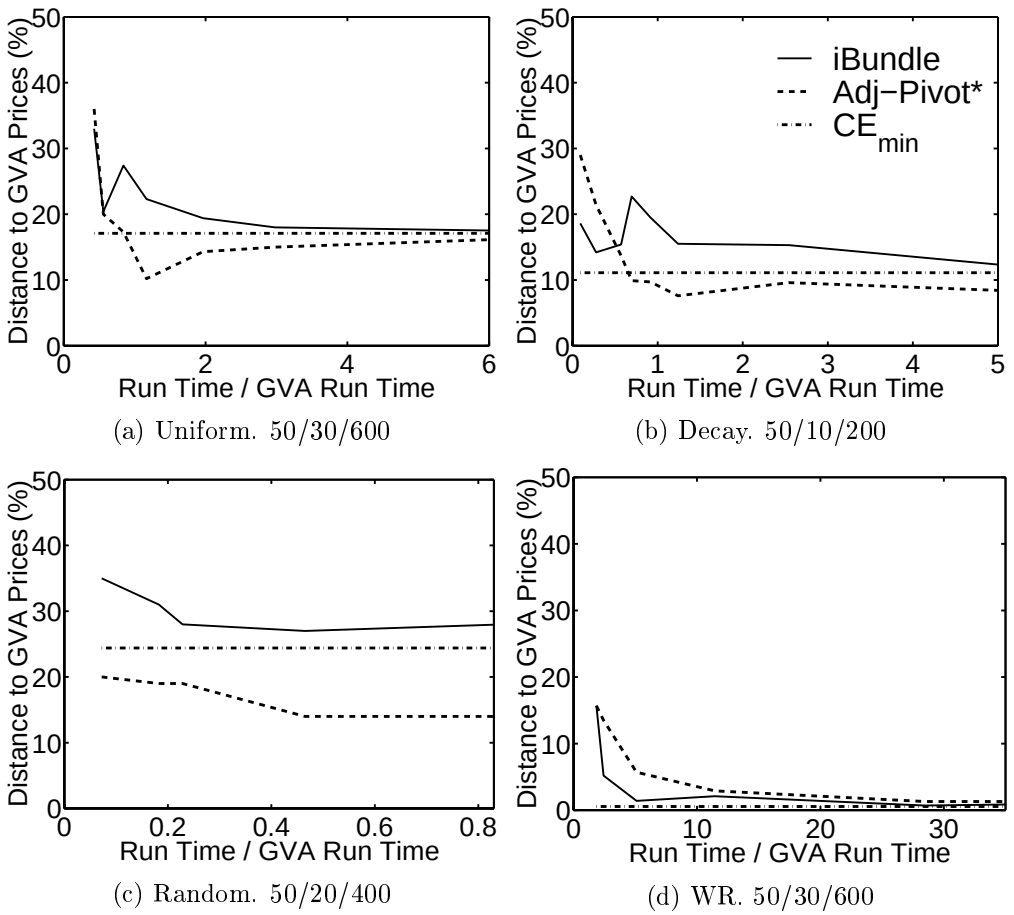
I also compute the fraction of all problems in which $\mathcal{D}(p_i, p_{\text{vick}}(i)) < 2\%$, to test the proportion of problems in which prices are approximately Vickrey. CE prices are equal to Vickrey payments in approximately 57% of problem instances; and *i*Bundle with both ADJUST* and ADJ-PIVOT* computes Vickrey payments in these problems, while *i*Bundle alone terminates with Vickrey payments in only around 38% of problem instances.

The minimal CE prices are close to GVA prices (average distance $< 2.5\%$) in problems PS 4–8, in which the agents in the optimal allocation also tend to be in the second-best allocations. In contrast, the minimal CE prices differ from the GVA payments by more than 5% in problems PS 1, 3, 9, 11 and 12, which are characterized by optimal allocations that are very different from second-best allocations, and agents with complementary demands for bundles (see Table 5.5).

As expected, additional analysis shows that the *Vickrey test* (Definition 6.10) is sufficient but not necessary for the adjusted prices to equal Vickrey payments. The *specificity* of the test was 100% (no false-positives), but its *sensitivity* was only 56% (some false-negatives); i.e. some cases in which the adjusted prices were in fact Vickrey payments were undetected.

The success of ADJ-PIVOT*, the approximate adjustment method that uses pivot allocations from earlier rounds of the auction, is confirmed in the results illustrated in Figure 7.3, for problems Decay, WR, Random, and Uniform. In Decay I set Sandholm's α parameter to 0.85. The distance to Vickrey payments is plotted against the run time of *i*Bundle with ADJ-PIVOT*, computed with respect to the time to solve the single-shot Generalized Vickrey auction (for the same winner-determination algorithm in both auctions). I varied the minimal bid increment to adjust the number of rounds in *i*Bundle, and study the distance between final prices and minimal CE and Vickrey payments as the allocative efficiency and correctness trends to 100%.

Minimal CE prices are equal to Vickrey payments in WR, because there is typically a single agent in the efficient allocation in this problem set. *i*Bundle with ADJ-PIVOT*



(a) Uniform. 50/30/600

(b) Decay. 50/10/200

(c) Random. 50/20/400

(d) WR. 50/30/600

Figure 7.3: Performance of *i*Bundle with price-adjustment ADJ-PIVOT* in problem sets Uniform, Decay, Random, and Weighted-random. The bid increment in *i*Bundle is adjusted to give different run times.

computes the Vickrey payments in these problems. It is interesting that ADJ-PIVOT* is able to compute prices *closer* to the Vickrey payments than the minimal CE prices in Decay and Random. Recall that the ADJUST* procedure minimizes each agent's payment separately, and can compute prices that are closer to the Vickrey payments than any single set of CE prices. Finally, notice that the minimal CE prices remain quite far from Vickrey payments in the Uniform problem set. The second-best allocations in Uniform are typically quite different from optimal allocations, and the agents-are-substitutes condition (Definition 6.3) often fails.

Although my focus is not on the auctioneer's winner-determination work, it is worth noting that the run time in the iterative auction is basically comparable to that in the

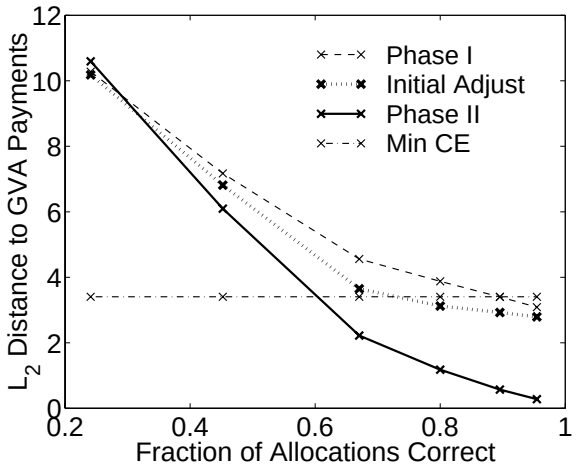


Figure 7.4: Distance to Vickrey Payments in PS 1–12.

GVA, if one considers the level of accuracy at which the performance of *i*Bundle and ADJ-PIVOT* levels out. The obvious exception here is in the weighted-random problem set, but this was actually the *easiest* problem to solve— the GVA solved the problem in an average time of 9.1 sec, compared with 362 sec, 1791 sec, and 138 sec for Decay, Random, and Uniform, on a 450 MHz Pentium.

7.7.2 Results II: *i*Bundle Extend&Adjust

*i*Bundle Extend&Adjust is tested on the same suite of problem instances; i.e. problems PS 1–12 and problems Decay, Weighted-random, Random and Uniform. Problem sizes (num agents, num items, num bundles) are set to (8, 50, 120) in Decay, (20, 50, 400) in Uniform, (8, 30, 80) in Random and (15, 50, 300) in Weighted-random. Decay parameter $\alpha = 0.85$. The results are averaged over 40 trials.

The distance between agent payments in the auction and GVA payments is measured with an L_2 norm, as $L_2(p_i, p_{\text{vick}}) = \left[\sum_i (p_i \Leftrightarrow p_{\text{vick}}(i))^2\right]^{1/2}$. I computed the average distance to Vickrey payments over the instances in which the auction terminates with the optimal allocation. As the bid increment gets small this fraction approaches 100%. This provides a more useful measure of distance than computing the average L_2 distance over all trials, including those in which the allocation is not efficient.

Figure 7.4 plots the distance between Vickrey payments and auction payments against the "correctness" of the auction, the fraction of instances in which the auction computes

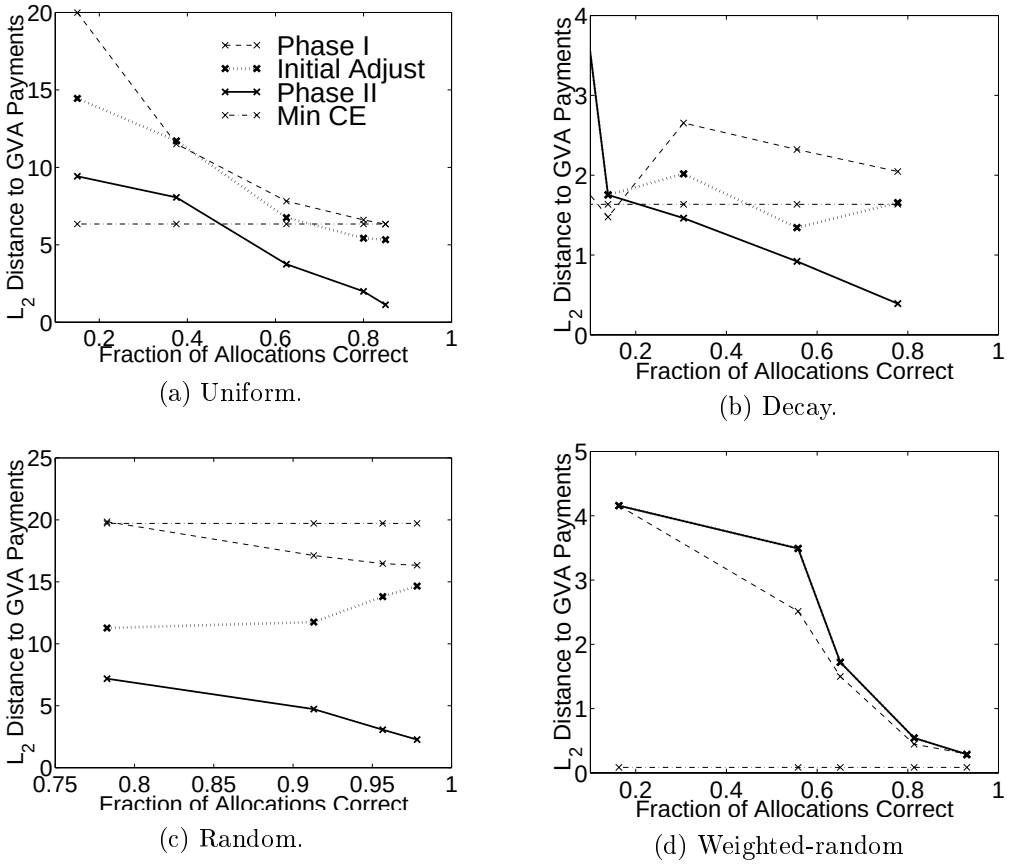
the efficient allocation, which approaches 100% as minimal bid increment $\epsilon \to 0$, for PS 1–12. The allocative efficiency in these experiments increases from around 90% at 23% correctness, to almost 100% at correctness of 65% and above, as the bid increment gets small. The figure plots the distance between Vickrey payments and: (1) prices at the end of PhaseI (*i*Bundle); (2) after the initial price adjust at the start of PhaseII (*i*Bundle and ADJUST*); (3) at the end of PhaseII (*i*Bundle, Extend&Adjust); and (4) minimal CE prices.

The adjusted prices in *i*Bundle Extend&Adjust do converge to the Vickrey payments as the minimal bid increment $\epsilon \to 0$ across this suite of problems, showing strong support for Conjecture 7.1 that the auction computes Vickrey payments with myopic best-response bidding strategies.

A quick computational analysis shows that the number of rounds on PhaseII is smaller than in PhaseI, although the auctioneer's computational problem is more difficult in each round of PhaseII. For example: at 95% correctness the average number of rounds in PhaseI is 149, compared to 18 in PhaseII; each round in PhaseI takes an average of 0.5s, compared to 2.8s in PhaseII; the agent valuation information provided at the end of PhaseI is 77%, and increases to 83% at the end of PhaseII, using the metric introduced in Section 5.3.1 in Chapter 5. Finally, approximately 1-in-3 agents receive a dummy agent during PhaseII.

Figure 7.5 illustrates the performance of the auction in problems Uniform, Decay, Random, and Weighted-random. In all problems allocative efficiency approaches 100% for small bid increments, and the distance to GVA payments approaches zero. The effect of PhaseII is quite significant, with the prices in the extended auction approaching Vickrey payments while the adjusted prices after ADJUST* but without PHASEII remaining close to the minimal CE prices. These results show further strong support for Conjecture 7.1 that *i*Bundle Extend&Adjust computes Vickrey payments in all CAP problems as the bid increment gets small, for myopic best-response bidding strategies.

It is worth noting that the auction implements the Vickrey outcome even in problems in which the outcome is *not* supported in any competitive equilibrium; notice that the distance between the minimal CE prices and the GVA payments is non-zero in all experiments. This is important because there is often no single set of CE prices that supports Vickrey payments.



(a) Uniform.

(b) Decay.

(c) Random.

(d) Weighted-random

Figure 7.5: Distance to Vickrey Payments in problems Uniform, Decay, Random and Weighted-random.

7.8 Discussion: PhaseI to PhaseII Transition

It is important that agents cannot identify the transition from PhaseI to PhaseII, because an agent's bids in PhaseII do not change either the final allocation or its own final payment. The only effect of an agent's bids in PhaseII is to reduce the final payment made by other agents. If it is costly to participate in the auction an agent would choose to drop out after PhaseI. In addition, there are opportunities for *collusion* between agents in PhaseII (just as the GVA itself is vulnerable to collusion).

Certainly, we must hide bids from dummy agents in PhaseI (or give the dummy agents false identities). Each agent only needs information about its own ask prices, and whether or not it is receiving a bundle in the provisional allocation. Agents do not need any information about the bids, prices, or allocations of other participants.

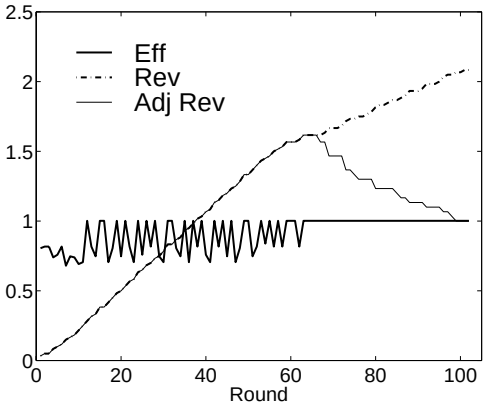
It is also important that agents cannot distinguish the competitive effects of bids from dummy agents from the competitive effects of bids from real agents. This is our reasoning for constructing dummy agents to mimic the real agents that compete for items in PhaseI of the auction.

Finally, another concern is that an agent cannot detect from the auctioneer's response time that we have moved from PhaseI into Phase II.

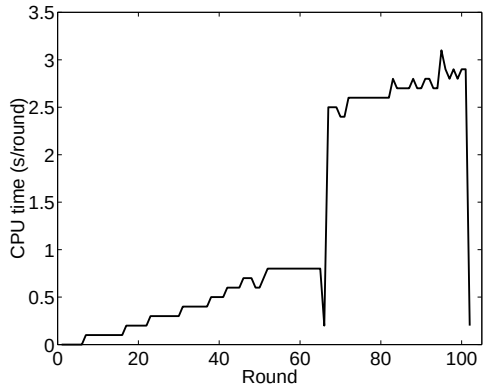
Figure 7.6 illustrates a typical run of *i*Bundle Extend&Adjust, in this case for a Uniform problem instance, with 25 goods, 10 agents and 150 bundles with strictly-positive value. Figure 7.6 (a) plots: (i) the efficiency of the allocation implemented during the auction; (ii) the revenue to the auctioneer, which increases monotonically across rounds; and (iii) the adjusted revenue, which falls towards the Vickrey payment during PhaseII. Notice that the efficiency of the allocation oscillates during PhaseI until it is locked-in, and that the final adjusted prices at the end of PhaseII are the Vickrey payments.

Figure 7.6 (b) plots the CPU time for the auctioneer in each round of the auction. Notice that it climbs slowly during PhaseI, as agents submit more bids in each round and the winner-determination problems get larger. The transition from PhaseI to II is quite apparent, at around iteration 70, when there is a jump in CPU time. The auctioneer must check Vickrey conditions at the end of every round during PhaseII.

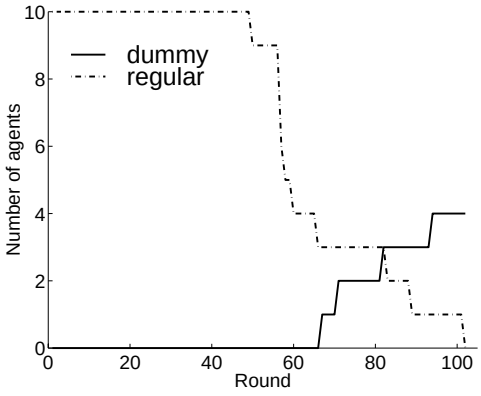
Figure 7.6 (c) plots the number of *active* agents and *dummy* agents during the auction. The number of active agents falls during PhaseI until all agents are in the provisional allocation, at which point the optimal allocation is found and the auction enters PhaseII.



(a) Performance.



(b) Auctioneer CPU time.



(c) Number of agents.

Figure 7.6: Uniform problem set example: 25 goods, 10 agents, 150 bundles.

During PhaseII the number of dummy agents increases as agents continue to drop out of the auction.

7.8.1 Unresolved Issues

There are a number of interesting areas for future work. I divide them into computational issues, and issues about agent incentives in *i*Bundle Extend&Adjust.

Computational

First, it should be possible to reduce the computational demands on the auctioneer. For example, it would be useful to allow the auctioneer to recompute the dependent agents in each round of PhaseII without explicitly computing the second-best revenue-maximizing allocations.

It would be interesting to investigate the performance of approximate methods such as ADJ-PIVOT*, which proved very effective when used to adjust prices immediately after *i*Bundle. The pivot method uses the cached provisional allocations computed during the auction to compute second-best allocations.

Agent Incentives

The separation of concerns between PhaseI and PhaseII in *i*Bundle Extend&Adjust is potentially dangerous. At the end of PhaseI the final allocation has been determined, and the only effect of bids from an agent during PhaseII is to reduce the final payment made by other agents. This separation is quite different from the Groves mechanisms, in which the agents submit their valuation functions up-front, and there is no sense of "what I am doing now is changing the allocation" and "what I am doing now is changing the prices". Of course, the situation in *i*Bundle Extend&Adjust is not completely decoupled. Bids during PhaseI do affect prices at the end of PhaseII, in addition to the allocation.

However, one concern is that the mechanism for potential *collusion*, which is a problem in the regular Groves mechanism, is quite explicit in *i*Bundle. For every $1 that agent 1 bids-up during PhaseII, one (or more) agents pay $1 less. To the extent to which the opportunities for collusion are so transparent it will be very important that there is a high-degree of uncertainty about the phase (I or II) of the auction.

It would be useful to investigate the incentive properties of *proxied* iterative mechanisms, and to understand the difficulty of manipulation in with myopic best-response proxy agents and consistency checks. An appropriate question to ask is the degree to which proxy agents "limit" the opportunities for manipulation in Proposition 7.2.

Finally, I would like to reduce the level of price discrimination in the auction. For example, in application to the allocation of a single item, the current auction maintains a separate ask price for each agent. In comparison, the English auction implements the Vickrey outcome with a single ask price which is the same to all agents.

Chapter 8

Bounded-Rational Compatible Auctions & Myopic Best-Response

This chapter introduces the idea of *bounded-rational compatibility*, which captures the concept of allowing an agent to implement an equilibrium bidding strategy without computing its exact preferences across all outcomes. Bounded-rational compatibility parallels the idea of strategy-proofness in auctions. In a strategy-proof auction an agent can compute its optimal strategy without any information about the preferences of other agents. In a bounded-rational compatible auction an agent can compute its equilibrium strategy (e.g. Bayesian-Nash, dominant-strategy, etc.) without complete information about its own preferences. In other words an approximate valuation function is sufficient to compute an optimal strategy. This useful in problems in which agents have limited computational resources, local valuation problems are hard, and there are many possible outcomes.

I also discuss the complexity of myopic best-response within the context of bounded-rational compatibility; i.e., consider the conditions that allow an agent compute its myopic best-response strategy without first evaluating its complete preferences over all outcomes, and consider the computational advantages to an agent. Clearly an agent can compute its set of best-response, or surplus-maximizing, bundles for a particular set of ask prices with suitable *bounds* on the values of each bundle, but what about best-response to a sequence of prices over the course of an iterative auction? An initial attempt is made to provide a *structural analysis* of the myopic best-response problem, to identify conditions in which myopic best-response is a *polynomial problem* while complete revelation is an *exponential problem.*

An explicit model of the bounded-rationality of agents can provide a new insight into computational mechanism design. In particular, we might take as a goal to maximize

allocative efficiency *given* the bounded-rationality of agents. In a sense this turns Russell's definition of "bounded-optimality" [Rus95] on its head. Instead of requiring that a bounded-rational agent takes the best decision possible given the constraints of its computational machine, let us design the environment of an agent— the mechanism —to extract the most value from the metadeliberation and resource-bounded computation of agents. Again, this brings us back to the concept of a bounded-rational compatible auction.

Experimental results presented in this chapter demonstrate that iterative auctions can compute more efficient allocations than sealed-bid auctions, for myopic best-response agent strategies and limits on agent computation. Well designed mechanisms can allow agents to avoid unnecessary computation *and* shift agent computation towards evaluating preferences towards local problems that are compatible with good system-wide solutions. An obvious example of good mechanism design is strategy-proofness, so that agents do not waste computational resources in deliberation about other agents. Bounded-rational compatibility captures this more subtle idea of providing agents with information to make good decisions about their own deliberation.

From a design perspective, one can imagine that bounded-rational compatibility introduces a new constraint to mechanism design, which can be taken in combination with other game-theoretic requirements such as incentive-compatibility and efficiency. The precise type of bounded-rational compatibility depends on the equilibrium concept adopted in analysis of a mechanism; e.g. dominant-strategy, Bayesian-Nash equilibrium etc. Essentially an agent must be able to compute its equilibrium concept with an approximate valuation function.

The formal definition of bounded-rational compatibility leads to a couple of metrics to quantify the degree of BRC in an auction for a particular problem and a particular model of agent bounded-rationality.

Experimental results are presented for two models of agent deliberation in equilibrium.

1. Costly computation and rational metadeliberation in a single-item allocation problem, comparing the properties of simple auction mechanisms.

2. Limited computation and myopic metadeliberation within a simple model of agent interaction, called "lazy deliberation and eager bidding" (LDEB).

The LDEB model provides a couple of metrics to quantify the degree of bounded-rational

compatibility of a mechanism. One metric, *bounded-efficiency*, measures allocative efficiency for different computation budgets. Another metric, *bounded-computation*, measures the amount of computation actually performed by agents for different computation budgets. In a bounded-rational compatible auction the bounded-computation peaks at a level below that required for agents to compute their complete preferences.

The outline of this chapter is as follows. First I introduce the agent valuation problem, define bounded-rational compatibility, and present a few examples. Then, I consider the complexity of myopic best-response in *i*Bundle, and characterize conditions on problem domains for which myopic best-response is *polynomial* while complete revelation is *exponential*. The last two sections present two sets of experimental results. Section 8.4 presents a comparison of simple auction models for a simple model of costly agent deliberation. Section 8.5 presents a computational study of auction models for a simple model of limited agent deliberation, in allocation problems with multiple items.

8.1 Agent Decision Problem

It is helpful to assume that an agent's decision problem can be separated into a *valuation problem*, to compute the value of different items, and a *bidding problem*, to compute an optimal bid. Each problem is well-defined in separation, for example the valuation problem can be solved with decision analysis tools and optimization methods that are independent of the particular auction, while the bidding problem can be solved with game-theoretic methods. Figure 8.1 illustrates this decision problem. The arrows show the flow of information.

Auction design can influence the effectiveness of an agent's deliberation on its local valuation problem because an agent can decide how to deliberate *dynamically* during the course of an auction, for example based on price information. Careful auction design can simplify an agent's valuation problem; auctions can allow an agent to *avoid* unnecessary computation on its valuation problem and bid with approximate valuations.

Before turning to valuation, note that the bidding problem can be hard, in particular when an agent with information about the bidding strategies of other agents can manipulate the outcome of the auction. Counter-speculation and game-theoretic reasoning is difficult. This is the critical role for strategy-proof mechanisms.

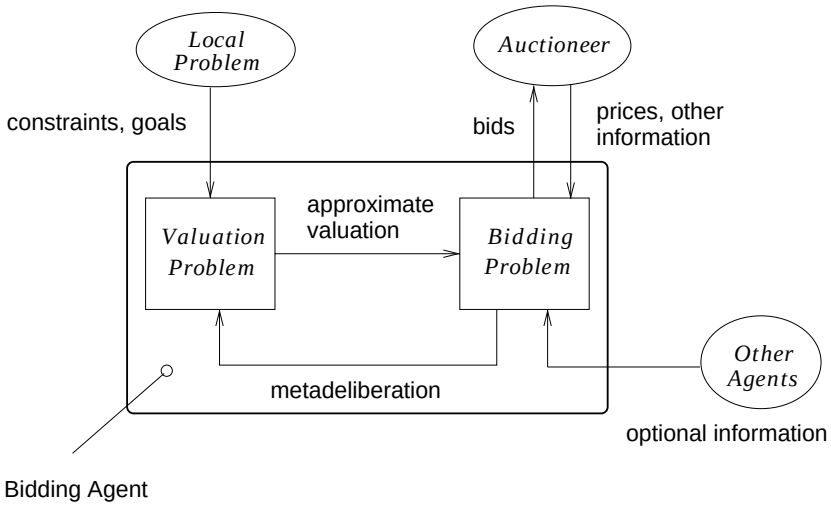


Figure 8.1: The agent decision problem.

As an example of a hard *valuation problem*, consider an auction for machine time in a multi-agent job-shop scheduling problem, in which each agent has jobs to schedule on a shared machine. To compute the value of a bundle of machine times an agent must compute its best schedule of jobs given the time specified in the bundle, which can be a NP-hard problem. Consider also an auction-based system for distributed task allocation where agents need to reformulate local plans to compute costs for performing additional tasks; or an auction-based system for allocating landing times at an airport, where the valuation problem is to compute the value of a time slot based on local constraints such as the availability of support and maintenance crews, gate availability, and costs for late arrival.

Let $v_i(S) \geq 0$ denote agent i's value for bundle $S \subset G$ of items, where G is a set of discrete items.

DEFINITION 8.1 [valuation problem] The valuation problem for agent i is to compute $v_i(S, \theta_i)$ for all bundles $S \subseteq G$, given preferences $\theta_i \in \Theta_i$, where Θ_i is the set of all possible types.

Recall that the *type* of an agent defines its preferences, in this case the value of an agent for all possible bundles of items. We can define an *approximate* valuation function in terms of stochastic information about an agent's type.

Let $\theta_{\text{app},i} \in \Delta(\Theta_i)$ denote a distribution over types, where $\Delta(\Theta_i)$ is the set of distributions on types. Similarly, $v_i(S, \theta_{\text{app},i}) \in \Delta(\mathbb{R})$ defines a distribution over the agent's value

for bundle S. An agent has approximate information about its valuation function if the value of one or more bundles is not completely defined:

DEFINITION 8.2 [approximate valuation] Agent i has an approximate valuation function, $v_i(S, \theta_{\text{app},i})$, if there is at least one bundle S' for which the value $v_i(S', \theta_{\text{app},i})$ remains uncertain.

Example approximate valuations in a combinatorial allocation problem include:

— lower bound, $\underline{v}_i(S')$, and upper bound, $\overline{v}_i(S')$, such that $\underline{v}_i(S') \leq v_i(S', \theta_i) < \overline{v}_i(S')$, for at least one bundle $S' \subseteq G$

— $v_i(S', \theta_i) = U(a, b)$, uniformly distributed between a and b, for some bundle $S' \subseteq G$

An agent's *metadeliberation* problem is to estimate the value of additional computation vs. acting, and to decide *what* to deliberate on and for *how long*. Russell and Wefald [RW91] present a general model of metareasoning, that considers computational actions explicitly within an agent's decision problem. A *bounded-rational* agent chooses a sequence of actions, computational or otherwise, to maximize expected utility, given models of computation and models of its world. The value of computation derives from the effect that the computation has on the actions that an agent takes in the world. Russell and Wefald develop myopic approximations to metareasoning and present an application to a stochastic decision problem.

A bounded-rational compatible auction allows an agent to perform useful metadeliberation. An agent with limited or costly computation and simple metadeliberation can avoid further deliberation about value when it knows that its bid is optimal for all possible values consistent with its approximate solution. An agent with *costly* computation, for example an opportunity cost associated with deliberation, considers the cost of deliberation and the expected value of the effect of further deliberation on the utility of its decision.

8.2 Bounded-Rational Compatible Auctions

A bounded-rational compatible (BRC) auction is an auction in which an agent can compute its optimal strategy, for a particular equilibrium concept, with an approximate valuation function and with *minimal information* about the other agents. We first introduced the term *bounded-rational compatible* auctions in Parkes et al. [PUF99]. It is important to

qualify the definition in terms of information about other agents, because we wish to identify auctions in which an agent can avoid valuation work *without* additional work modeling the other agents. To understand this, consider the following example.

Example. An agent in the Vickrey auction without any information about the preferences or strategies of other agents cannot compute its optimal strategy without complete information about its own valuation problem, because the optimal strategy of an uninformed agent is to bid its complete and accurate valuation function. However, an *informed* agent in the Vickrey auction *can* compute its optimal strategy with approximate value information. For example if an agent knows that the highest bid from another agent is \$10 then it can bid optimally with a lower bound on its own value of \$12; i.e. a bid of $b = \$12$ will maximize utility.

A natural way to handle information in the characterization of bounded-rational compatible auctions is to require that an agent can compute its equilibrium strategy without any *additional* information about the problems of other agents beyond that which is required to compute the optimal strategy with complete information about the agent's own problem.

We define the concept of an *information-restricted* agent:

DEFINITION 8.3 [information-restricted] An agent i is information restricted if it has the minimal amount of information about the other agents needed to compute its equilibrium strategy with complete information about its own preferences.

In other words, an information-restricted agent has no more information than is minimally required by the solution concept in the mechanism. For example:

— an information-restricted agent in a dominant-strategy equilibrium has *no information* about the other agents, because it requires no information to compute its dominant strategy.

— an information-restricted agent in a *Bayesian-Nash* equilibrium has the distributional information about agent preferences required to compute the expected-utility maximizing strategy in the Bayesian-Nash equilibrium.

With this concept we can define a bounded-rational compatible auction. The equilibrium concept undefined in the general definition.

DEFINITION 8.4 [bounded-rational compatible] An auction is bounded-rational compatible (BRC) if an *information-restricted* agent can compute its equilibrium strategy with approximate information about its valuation problem in one or more non-trivial problems.

In other words, in a BRC auction an agent can compute its equilibrium strategy with incomplete information about its preferences and minimal information about the other agents. This definition precludes an agent from compensating for less information about its own problem by collecting information about the problems of the other agents.

We also exclude *trivial problems* from consideration:

DEFINITION 8.5 [trivial problem] An instance of mechanism $\mathcal{M}$ for agent i, as defined with preferences $\theta_{-i} = (\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \theta_I)$ of the other agents, is *trivial* if the outcome implemented in equilibrium $f(\theta_i, \theta_{-i})$ is the same irrespective of the type $\theta_i \in \Theta_i$ of agent i.

A problem is trivial for agent i if the same outcome would be implemented by the mechanism for all possible preferences of agent i, even when agent i has complete and accurate information about its preferences.

Example. Consider a second-price sealed-bid auction in which agent 1 is the only agent. This is a trivial problem because the agent will win the item for \$0 whatever its bid.

We exclude trivial problems from the definition of BRC because an agent can trivially compute its optimal strategy with approximate information about its preferences in a trivial problem (so long as it knows the problem is trivial!), because it can simply follow a strategy for *any* possible set of preferences.

The precise interpretation of equilibrium in BRC is left undefined, and depends on the analysis method adopted for a mechanism. Special cases of bounded-rational compatibility, for particular equilibrium concepts, include:

DEFINITION 8.6 [dominant BRC] An auction is dominant-strategy BRC if an information-restricted agent can compute its *dominant strategy* with approximate information about its valuation problem, in one or more non-trivial problems.

DEFINITION 8.7 [myopic BRC] An iterative, price-directed, auction is myopic BRC if an information-restricted agent can implement its *myopic best-response* strategy *in every*

round with approximate information about its valuation problem, in one or more non-trivial problems.

Myopic bounded-rational compatibility requires that an agent computes its optimal strategy in every round of the auction with an approximate valuation function, not just in a single round.

Notice also that I choose to be a little loose with the use of "equilibrium" here, as myopic best-response is not actually an equilibrium strategy for rational agents in an iterative auction such as *i*Bundle; rather it is an equilibrium strategy for myopic agents that view the current round as the last round of the auction and take prices as given. Of course myopic best-response *is* a (Bayesian-Nash) equilibrium of iterative Vickrey auctions, in which case I would say that such an auction is *Bayesian-Nash* bounded-rational compatible.

8.2.1 Illustrative Examples

In this section I present some examples of BRC analysis, to the English and Vickrey auctions, and a Posted-price market.

English Auction

PROPOSITION 8.1 *The English auction is myopic bounded-rational compatible.*

PROOF. Consider the English auction, an ascending-price auction for a single item, with myopic best-response agent strategies. An agent that knows its value v can compute its optimal myopic best-response strategy without any information about the bids from other agents, i.e. $b^*(p) = p$ if $p \leq v$, and drop out of the auction otherwise. An agent can also compute its optimal strategy with an approximate valuation, e.g. bounds $\underline{v} \leq v \leq \overline{v}$, in some problems. For example, if the highest outside bid $\overline{b} = v \Leftrightarrow \Delta$, then the agent can compute its optimal strategy (bid at every price) with lower bound $\underline{v} > v \Leftrightarrow \Delta$. ∎

Consider the simple example in Figure 8.2. Agents 3 and 5 will bid the ask price up to just above agent 5's lower bound, the second-highest lower bound. At this price, agents 1 and 2 can drop out of the auction without computing their exact values, while agents 3, 4 and 5 must perform further deliberation. This bidding problem demonstrates the BRC property of the English auction, in this example, for agents 1 and 2.

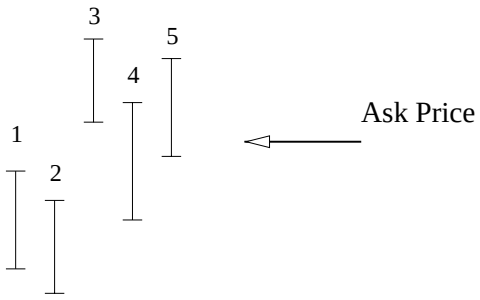


Figure 8.2: Example scenario in the English auction.

In fact, the English auction also terminates with the Vickrey payment for small enough bid increments, and myopic best-response is a Bayesian-Nash equilibrium for agents; i.e., given that every agent is following a myopic best-response strategy (for some value) then an agent's rational sequential strategy is also myopic best-response. This was discussed extensively in Chapter 7.

The following result follows:

PROPOSITION 8.2 *The English auction is Bayesian-Nash bounded-rational compatible.*

This is straightforward to show, given that the auction is myopic bounded-rational compatible, and given that myopic best-response is a Bayesian-Nash equilibrium.

The Vickrey Auction

PROPOSITION 8.3 *The Vickrey auction is not dominant bounded-rational compatible.*

PROOF. An agent that knows its value v can compute its dominant strategy without any information about the bids from other agents, i.e. $s^* = v$. Therefore, in order to be *dominant* BRC the Vickrey auction must allow an agent to compute its dominant strategy without any information about the bids from other agents, and approximate value information for its own problem. But its dominant strategy in this uninformed case is truthful and complete revelation of its value information, which presents a conflict with the demands of BRC. ∎

Posted-Price Market

PROPOSITION 8.4 *A posted-price market is* dominant *BRC.*

PROOF. A posted-price market with infinite supply, or with an *exclusive* take-it or leave-it offer to an agent, is not a situation of strategic interaction. There is no relevant information about the other agents, and the optimal strategy— also a dominant strategy —is to accept the price on an item if the price is below an agent's value. An agent can compute this optimal strategy with approximate information about its value, for example with a lower-bound that is above the ask price. ▮

8.2.2 Preliminary Theoretical Results

It is useful to outline some possibility and impossibility results for mechanism design, once bounded-rational compatibility is introduced into the mix of desirable auction properties.

THEOREM 8.1 (possibility). *The English Auction is efficient and Bayesian-Nash bounded-rational compatible for the single-item allocation problem.*

PROOF. Myopic best-response is a Bayesian-Nash equilibrium of the English auction, i.e. the myopic best-response is sequentially rational for an agent given myopic best-response by every other agent (see Section 7.4). The English auction is efficient with this strategy, and we have already shown an example of how myopic best-response can be implemented with an approximate valuation function. ▮

THEOREM 8.2 (possibility). i*Bundle is efficient and myopic bounded-rational compatible for the combinatorial allocation problem.*

PROOF. *i*Bundle is efficient with myopic best-response strategies, and it is quite straightforward to construct non-trivial problems in which at least one agent can implement its myopic best-response strategy in each round without complete information about its valuation function. ▮

For example, in the simple problem in Section 3.2.2 an agent can bid for its value maximizing bundle with a partial-ordering over bundles.

THEOREM 8.3 (impossibility). *No strategy-proof (direct-revelation) auction can be dominant strategy bounded-rational compatible and efficient.*

PROOF. Dominant strategy bounded-rational compatibility requires that an agent can compute its equilibrium strategy (in this case its dominant strategy) with an approximate valuation function and with no information about the preferences or strategies of other agents. A single-shot direct-revelation mechanism cannot be efficient unless every agent submits complete and accurate information about its valuation function in every problem. Any approximate information, for example specifying an incorrect value for bundle S' can be exploited by constructing a non-trivial problem in which the values of the other agents for particular bundles make the error in the value of bundle S' cause the selection of an inefficient allocation. An uninformed agent with an approximate valuation function cannot submit complete and accurate information about its valuation function in every problem. ∎

8.2.3 Discussion

Although the definition of bounded-rational compatibility requires only a *single* non-trivial problem in which the agent can compute its optimal strategy with approximate information about its valuation problem the definition is successful in separating direct-revelation and iterative auction designs. Bounded-rational compatibility also captures other market mechanisms, such as posted-price mechanisms, and demonstrates that there is often a trade-off between allocative-efficiency and bounded-rational compatibility. Of course, iterative auctions such as *i*Bundle are interesting because they can be BRC *and* efficient.

8.2.4 Approximation-Proofness

The strategy-proofness of a mechanism can break when agents have approximate valuations [San96]. An agent that is informed about the bids that another agent will place can optimize its local computation about value, and submit more useful bids in the auction.

For example, the Vickrey auction is *not* strategy-proof for an agent with an uncertain value for its item and costly computation. The agent can avoid costly computation if it has access to free information about the other agent.

PROPOSITION 8.5 *A bounded-rational agent with uncertain values and limited computation can improve its expected utility with information about the bids from other agents in a strategy-proof auction.*

For example, if an agent has uncertain values on two items, but can only refine its value for one item, it is helpful to know the likely prices on the items. In this way, the bounded-rationality of agents can break the strategy-proofness of an auction.

However, so long as a mechanism satisfies *approximation-proofness*, then this loss in full strategy-proofness is not a problem. The concept of approximation-proofness parallels that of strategy-proofness, and states that an agent's dominant strategy is to report truthful information (or statistics) about its approximate valuation function. As before $\Delta(\Theta)_i$ denotes the set of possible approximations, given a set of possible types Θ_i for agent i. Let $\mu(\Delta(\Theta)_i)$ denote the set of possible statistics, given all possible approximations. This defines the *strategy-space* available to an agent in a direct-revelation approximation-proof mechanism (see Section 2.3 for an introduction to direct-revelation mechanisms). I define approximation-proofness as follows:

DEFINITION 8.8 [approximation-proof] A direct-revelation mechanism $\mathcal{M} = (\mu(\Delta(\Theta)_1), \ldots, \mu(\Delta(\Theta)_I), g(\cdot))$ is approximation-proof for statistics $\mu(\theta_{\text{app},i})$, if the dominant-strategy equilibrium for agent i is to report truthful statistics about its approximate type information, i.e. $s_i(\theta_{\text{app},i}) = \mu(\theta_{\text{app},i})$ for all possible approximate types.

In other words, a mechanism is approximation-proof for statistics $\mu(\theta_{\text{app},i})$, such as *mean*, *upper-* and *lower-bounds*, etc., if an agent's dominant strategy is to reveal this information about its approximation truthfully.

A special case is that the mechanism simply allows the agent to provide its complete approximation, for example upper- and lower- bounds on the value of every bundle, and truthful revelation of this information is an agent's dominant strategy.

Example. The single-item Vickrey auction, in which an agent can report a single value for the item, is approximation-proof for mean statistics, with an expected utility maximizing agent with a quasi-linear utility function. Given approximate information $v_i(\theta_{\text{app},i})$ the agent's dominant strategy is to report its true expected value. To understand this, notice that the agent wants to buy the item whenever the highest bid from another

agent is less than its expected value.

Approximation-proofness does not imply that a mechanism is strategy-proof, it is a weaker condition. An agent in an approximation-proof mechanism might continue to collection information about other agents, and model the other agents, in equilibrium. However, in an approximation-proof mechanism this loss in strategy-proofness is not necessarily a *bad* loss in strategy-proofness. In an approximation-proof auction it is *beneficial* for an agent to allocate its computational resources to maximize the effectiveness with which its final valuation function provides accurate and useful information to the mechanism. With approximation-proofness this information is used to perform better metadeliberation on the agent's own valuation problem, and an agent will still reveal truthful information to the mechanism. Of course, we would prefer an iterative auction that can guide agent metadeliberation, and provide information to guide an agent's valuation deliberation for "free" (this is the idea in bounded-rational compatible auctions).

I leave the design of approximation-proof mechanisms for combinatorial problems as an important open problem. Note, for example, that the GVA is *not* approximation-proof with expected-value information. The property that holds in the single-item Vickrey auction requires linearity, which does not hold in the combinatorial allocation problem and the GVA. By the revelation principle, we could simply look for approximation-proof direct-revelation mechanisms, that allow an agent to report *complete information about its approximation* to the mechanism. Perhaps more interesting is the design of approximation-proof mechanisms for aggregate statistics on an agent's approximate information about its preferences, such that truthful information of this statistic is a dominant strategy.

8.3 Complexity of Myopic Best Response

A running assumption in my dissertation can be stated as follows:

CLAIM 8.1 *iBundle solves* realistic problems *with less information and less agent computation than the GVA.*

In hard problem instances we will need complete information revelation from agents to compute and verify an optimal allocation, i.e. the performance of *i*Bundle is just as bad

as the GVA in the worst case.

The claim is that despite the same worst-case performance, the average-case performance of *i*Bundle is good, in that we solve easy instances with as little information as possible. Back in Chapter 3 I introduced a few examples that can be solved with incomplete information from agents. But, what about the computation required by an agent to provide this incomplete information?

Leaving aside whether or not the *sequential* complexity of myopic best-response is harder or easier than direct-revelation complexity, let us consider the simpler problem of responding to a single price vector.

DEFINITION 8.9 [best-response problem] Compute set of bundles, BR($\mathbf{p}$), to solve:

$$\max_S v(S) \Leftrightarrow p(S)$$

The following section considers conditions on an agent's local problem, valuation problem, and approximation algorithm for best-response to be easy while complete revelation is hard.

8.3.1 Structural Analysis

One approach is to characterize the structure of an agent's valuation problem, approximate valuation complexity, and exact valuation complexity, in which best-case myopic best-response is polynomial-time computable (in the number of items) while worst-case complete revelation is exponential-time computable (in the number of items).

Consider the following cases, in which myopic best-response is computable in best-case polynomial time, while complete revelation remains hard:

(a) A polynomial-time **approximate valuation algorithm**, an exponential-time exact valuation problem, and a polynomial number of interesting bundles.

(b) A polynomial-time **approximate inference capability** (e.g. if $v(S_1) = x$ then $v(S') < x$, for all $S' \prec S_1$), with exponential inference power, a polynomial algorithm to compute the exact value of a single bundle, and an exponential number of interesting bundles.

In case (a), the agent can compute its complete approximate valuation function in polynomial time, computing an approximate value for each bundles in polynomial time on a polynomial number of interesting bundles. It remains an exponential-time problem to compute the agent's exact valuation function, because it must do an exponential amount of work on each of a polynomial number of bundles.

In case (b), the agent can compute its complete approximate valuation function in polynomial time, because it can compute the approximate value of an exponential number of bundles (this is what is meant by the "exponential power" statement) in polynomial time, based on the exact value of a single bundle that it can compute in polynomial time. This approximate inference capability can be applied to a number of "seed" bundles, until the agent has enough information to compute its best-response. It remains an exponential-time problem to compute the agent's exact valuation function, because it must do a polynomial amount of work on each of an exponential number of bundles.

Of course, this analysis is simplistic and ignores the subtle question of whether the approximate information allows an agent to compute its best-response. Factors that should boost the ability of an agent to compute its myopic best-response with approximate information and limited deliberation are:

— high variance in values for different bundles with similar prices, or ask prices that are very different in structure from values.

— a small number of interesting bundles to enable single-bundle approximation methods without inference across bundles

— a structured local problem to allow approximate value inferences across bundles

— linear prices instead of non-linear prices on bundles

The results in the next couple of sections present an experimental comparison between the computational and economic properties of different auction mechanisms with a simple model of a bounded-rational agent. The results demonstrate the effect that auction design can have on the efficiency of an allocation. Iterative auctions allow agents to follow strategies with less value computation *and* can lead to more efficient allocations because agents use deliberation to refine values on important items and/or bundles as feedback is provided by the mechanism about the bids and preferences of other agents.

8.4 Costly Deliberation and Single-Item Allocation

The first set of experiments consider costly agent deliberation in a single-item allocation problem. I compare the performance of three simple auction models: a sealed-bid auction, an ascending-price auction, and a posted-price market.

An agent's *metadeliberation problem* is to determine how much deliberation to perform before placing a bid. The decision is a tradeoff between reducing uncertainty about the value of the good so that the bid is accurate, and avoiding the cost of deliberation. Given the model of an agent's valuation problem and decision procedure we derive optimal metadeliberation strategies for agents. The key observation is that the value of deliberation is derived from the effect of deliberation on an agent's bid. Deliberation can only be worthwhile when it changes an agent's bid and expected utility from the decision is greater than cost. To the best of my knowledge, this was the first model of normative agent metadeliberation within an iterative auction [Par99].

An agent's optimal metadeliberation strategy *does* depend on the bids that other agents will make, even in incentive compatible auctions (unlike an agent's optimal bidding strategy). For example, an agent should never deliberate about its value for a good if its current upper bound on value is less than the ask price, because further deliberation can never cause the agent to accept the price. Metadeliberation is hard because of uncertainty about the bids of other agent and the outcome of additional deliberation.

I describe normative metadeliberation strategies for risk-neutral agents, who receive utility $v_i \Leftrightarrow p$ for purchasing a good at price p.

8.4.1 Model of Agent Bounded-Rationality

In Parkes [Par99] I proposed a simple model for the valuation problem of an agent, and derived myopic metadeliberation strategies and bidding strategies in different auction mechanisms. I do not expect the valuation problems and decision procedures of real agents (or real experts) to have characteristics that match the precise assumptions (e.g. distributional assumptions) of the model. However, I believe that the same qualitative effect will be observed with alternative models of agent deliberation and hard valuation problems.

The model of approximate valuation matches some of the properties of standard algorithmic techniques for solving hard optimization problems, such as Lagrangian relaxation,

depth-first search, and branch-and-bound. Furthermore, the model supports a mode of interaction between people and software bidding agents that is provided in some current on-line auctions [PUF99].

Every agent i has an unknown true value v_i for a good, and maintains a lower bound $\underline{v}_i$ and upper bound $\overline{v}_i$ on its value, written $[\underline{v}, \overline{v}_i]$. Agent i believes that its true value is uniformly distributed between its bounds, $v_i \sim U(\underline{v}_i, \overline{v}_o)$. Given this belief the expected value for the good is $\hat{v}_i = (\underline{v}_i + \overline{v}_i)/2$. As an agent deliberates its bounds are refined and its belief about the value of the good changes, with expected value $\hat{v}$ converging to v over time.

Let $\Delta_i = \overline{v}_i \Leftrightarrow \underline{v}_i$ denote an agent's current uncertainty about the value of the good. Agents have a deliberation procedure that adjusts the bounds on value, reducing uncertainty by a multiplicative factor α, where $0 < \alpha < 1$. The new bounds are $\alpha\Delta_i$ apart, and consistent with the current bounds (but not necessarily adjusted symmetrically). For a small α the uncertainty is reduced by a large amount, and we refer to $(1 \Leftrightarrow \alpha)$ as the *computational effectiveness* of an agent's deliberation procedure.

An agent believes that the new expected value $\hat{v}_i'$ for the good after deliberation will be uniformly distributed $\hat{v}_i' \sim U(\underline{v}_i + \alpha\Delta_i/2, \overline{v}_i \Leftrightarrow \alpha\Delta_i/2)$, such that the new bounds are consistent with the current bounds.[1]

Finally, there is a cost C associated with each deliberation step performed by an agent, measured in units of payment; i.e. an agent's utility for value v and cost C is $v \Leftrightarrow C$.

8.4.2 Auction Models

I compare the following auctions: second-price sealed-bid [SB], sequential posted-price [PP], and ascending-price [AP]. The [PP] and [AP] auctions are BRC (dominant and myopic), the [SB] auction is not BRC.

The auctions place different information requirements on the auctioneer: the [PP] auction requires a well-informed auctioneer for good performance, because the ask price is critical; while the [AP] and [SB] auctions set the price dynamically from bids received during the auction.

[1] This belief that the new bounds are uniform with respect to the current bounds is inconsistent with the prior belief that the actual value of the item is uniformly distributed between the bounds. To support a uniform value for the item over a sequence of deliberations the distribution for the value in the next round must put more weight on values towards the center of the range. In simulation this distribution is carefully computed to maintain a uniform prior for the true value after any sequence of deliberations.

Second-price Sealed-bid [SB]

Actions: `bid`.

In [SB] the auctioneer accepts bids b_i from agent i, and then closes the auction. The item is sold to the agent that submits the highest bid, for a price equal to the second-highest bid (or \$0 if only one bid is received). This is the Vickrey auction.

Sequential Posted-price [PP]

Actions: `accept, reject`.

In [PP] the auctioneer offers the item to each agent sequentially, for price p. The item is sold to the first agent that `accept`s the price. The item is not sold if every agent `reject`s the price.

In simulation I optimize the ask price off-line to maximize revenue, given agent bidding and deliberation strategies. This is equivalent to assuming a well-informed auctioneer.

Ascending-price [AP]

Actions: `register, leave, bid, deliberate`.

I use a variant on the standard ascending-price (English) auction that is designed to simplify the analysis of deliberation and bidding strategies, without changing the performance of the auction for rational agents. The auction includes a nominal charge for remaining in the auction, and allows agents to explicitly state that they will `leave` the auction.

Initially all agents `register` with the auctioneer. Then, agents can place bids until they `leave`. Bids indicate the *maximum* that an agent is currently prepared to pay for the item. A bid b_i is accepted if $b_i \geq p$, and the ask price is increased to ϵ above the second-highest bid received, for some bid increment $\epsilon > 0$. Agents can place new bids at any time, and leave the auction at any time. Agents are charged a nominal participation fee to remain in the auction, unless they hold the highest bid.

After a period of time without new bids, or without an agent leaving the auction, the auctioneer announces the number of active agents in the auction, and the auction enters a "going going gone" phase. An agent must `bid`, `leave` or `deliberate` to keep the auction open. When an agent decides to deliberate it sends a `deliberate` message to the auctioneer. The auctioneer then gives the agent time to deliberate, and then expects either

a `bid` or a `leave` message. The agent is dropped from the auction if it does not respond. Finally, the auction will close, with the item sold to the agent with the highest bid for the value of the second-highest bid received.

An alternative solution to prevent agents free-riding off the deliberation of other agents is to choose an agent at random in each round of the auction, and requests that the agent places a bid or leaves the auction.

8.4.3 Metadeliberation and Bidding Strategies

In this section I briefly describe optimal agent deliberation and bidding strategies in each auction. The full mathematical analysis of an agent's optimal metadeliberation strategy in [PP] is presented in the Appendix of this chapter. The optimal metadeliberation strategy in [PP] is also relevant in [AP]. The value of deliberation is computed within Russell & Wefald's [RW91] decision-theoretic framework.

Second-price Sealed-bid [SB]

The optimal bid, b^*, for a risk-neutral agent with approximate value $[\underline{v}_i, \overline{v}_i]$ that will perform no further deliberation, is:

$$b_i^* = \hat{v}_i = \frac{\underline{v}_i + \overline{v}_i}{2}$$

To decide how long to deliberate before bidding an agent needs to know the utility of its bid, but this depends on the bids placed by other agents— information not available to an uninformed agent. The agent can assume that the highest outside bid, p, is equal to its current approximate value for the item, $p = \hat{v}_i$, but this will lead the agent to perform too much deliberation. Similarly, if an agent assumes that $p = \underline{v}$ or $p = \overline{v}$ it will tend to perform too little deliberation.

I assume that agents can compute a *symmetric pure-strategy Nash equilibrium*, where every agent performs the same number of deliberations. The equilibrium number of deliberations are computed off-line (to simulate long-term learning), so that the total cost of deliberation is just less than the expected utility to the agent that wins the auction. This provides a best-case number of deliberations for a symmetric pure Nash equilibrium of metadeliberation decisions.

Value bounds	Deliberation bounds	Expected value	Optimal action for price $p = 5$
[6,8]	[6.8,7.2]	7	`accept`
[4,7]	[5.05,5.95]	5.5	`accept`
[3.5,6.5]	[4.55,5.45]	5	`deliberate`
[3,8]	[4.5,6.5]	5.5	`deliberate`
[4.5,5.3]	[4.9,4.9]	4.9	`reject`
[2,4]	[2.8,3.2]	3	`reject`

Table 8.1: Deliberation bounds for an agent with $\alpha = 0.5$ and $C = 0.05$ and different approximate values. The optimal action (`deliberate`, `accept` or `reject`) is computed for ask price $p = 5$.

Sequential Posted-price [PP]

The optimal bidding strategy depends on the relation between the ask price p and the agent's belief about its expected value $\hat{v}_i$:

$$b^* = \begin{cases} \texttt{accept} & \text{, if } p < \hat{v}_i \\ \texttt{reject} & \text{, otherwise.} \end{cases}$$

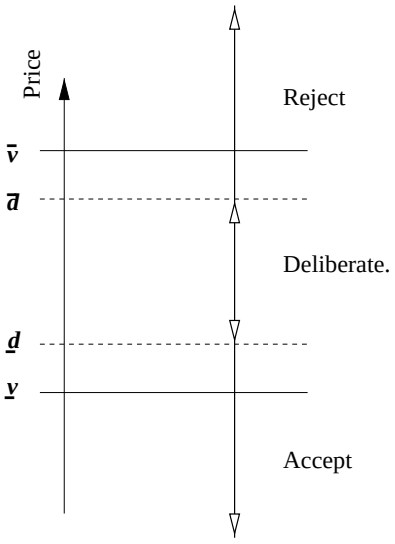
In [PP] the utility of a bid depends on the expected value $\hat{v}_i$, but not on the bids placed by other agents. The optimal deliberation strategy depends on the price, an agent's approximation, and the deliberation procedure parameters α and C. An agent will deliberate while the ask price is "close" to its current approximate value $\hat{v}_i$, where close is defined with *deliberation bounds*:

DEFINITION 8.10 [deliberation bounds] An agent should `deliberate` when the ask price p is $\underline{d}_i < p < \overline{d}_i$, where $\underline{d}_i$ and $\overline{d}_i$ are lower upper deliberation bounds, and depend on $[\underline{v}_i, \overline{v}_i], \alpha, C$.

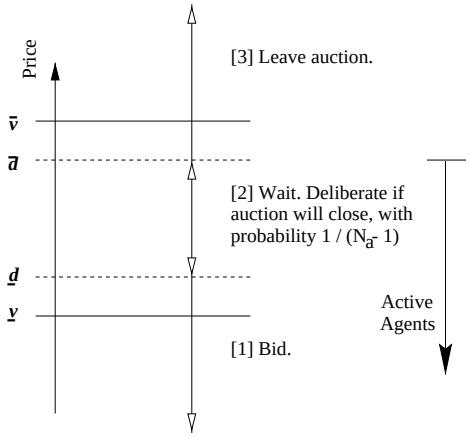
I derive an analytic formula for the deliberation bounds in the Appendix. Table 8.1 presents bounds for an agent with $\alpha = 0.5$ and $C = 0.05$, and different approximate values. The table also records the optimal action for ask price $p = 5$ in each case, i.e. `deliberate`, `accept` or `reject`. The deliberation bounds are always tighter than the value bounds. An agent should never deliberate when the ask price is outside its value bounds because deliberation cannot change its bid. Deliberation is more useful as the price is closer to expected value, as uncertainty increases, and as deliberation effectiveness increases, because deliberation is more likely to change an agent's bid in each case.

Figure 8.3 (a) illustrates the combined bidding and deliberation strategy for an agent in [PP]. The strategy is as follows: when the ask price is between the bounds on deliberation,

`deliberate`. Otherwise: if the ask price is less than $\hat{v}_i$ (expected value of the item), `accept`, and if the ask price is greater than $\hat{v}_i$ then `reject`.



(a) Posted-Price market [PP].



(b) Ascending Price auction [AP].

Figure 8.3: Optimal bidding and deliberation strategies in each auction. The optimal action depends on the ask price, p, and the deliberation bounds. Agents are more bounded-rational from level [4] to [1].

Ascending-price [AP]

The optimal myopic bidding strategy for a bounded-rational agent in [AP] depends on the relation between the current ask price and an agent's *deliberation bounds*.

$$b^* = \begin{cases} \texttt{bid} & \text{, if } p < \hat{d}_i \\ \texttt{leave} & \text{, if } p > \hat{d}_i \\ \texttt{wait} & \text{, otherwise.} \end{cases}$$

The strategy is equivalent to accepting any ask price less than $\underline{d}_i$ and rejecting any ask price greater than $\overline{d}_i$, which is the optimal strategy in [PP]. However, when the ask price is between the bounds an agent will now `wait`, and only deliberate when it must deliberate and bid to keep the auction open. Agents that either `wait` or `bid` remain *active*.

This is a *myopic* metadeliberation strategy because the agents select deliberation and auction actions in equilibrium *at the current prices*, and ignore the possibility that prices might continue to increase and that an agent will not necessarily when an item at the current ask price. In particular, the price might increase so that $p > \overline{d}_i$, in which case the

agent is better to `leave` the auction than `deliberate`. This myopia is somewhat relaxed because I assume that agents will try to "free-ride" off the deliberation of other agents, and only deliberate when absolutely necessary (e.g. to prevent the auction closing).

In the going-going-gone phase of the auction, an agent with bounds $\underline{d_i} < p < \overline{d_i}$, will wait if another agent will deliberate and bid, and only deliberate and bid to prevent the auction from closing.[2] I call this the "waiting game". Agents try to free-ride off the deliberation of other agents. All deterministic Nash equilibrium solutions have a single agent that `deliberates`, while all other agents `wait`: if no agent `deliberates` then it is a rational unilateral deviation for any single agent to `deliberate`, and if more than one agent `deliberates` that it is a rational unilateral deviation for one of those agents to `wait`.

In the unique *mixed* Nash equilibrium (also symmetric) of the waiting game, with agents that randomize among strategies, every agent `deliberates` with probability $1/(N_a \Leftrightarrow 1)$, where N_a is the number of active agents. The denominator is $N_a \Leftrightarrow 1$ because the agent that is currently holding the highest bid will not deliberate. Uninformed agents can implement this strategy because the auction announces the number of active agents, N_a, at the start of every round. The agents do not need to know the strategies or utility functions of other agents. Figure 8.3 (b) illustrates the combined bidding and deliberation strategy of an agent in [AP]. The Nash equilibrium allows agents to deliberate sequentially. Agents exchange information about their local problems through the ask price as they deliberate and bid.

I assume in simulation that agents implement this outcome, and simulate the mixed Nash equilibrium by choosing an active agents at random to deliberate. An agent that is selected deliberates until the ask price is outside its deliberation bounds, when it either bids or leaves the auction.

8.4.4 Experimental Results: Costly Computation

I compare the performance of each auction in terms of the allocative efficiency.

In each trial a value for the item is assigned to each agent from the same distribution, $v_i = U(0, 10)$, uniform between 0 and 10. Every agent has initial bounds on value, $[\underline{v_i}, \overline{v_i}] = [0, 10]$ (i.e. ignorance). Each auction is tested with four different agent bounded-rational levels, as summarized in Table 8.2, and with between $|\mathcal{I}| = 5$ and $|\mathcal{I}| = 100$ agents. In

[2] We assume that the auction allows enough time for an agent to deliberate later in the auction.

each experiment all agents have the same deliberation effectiveness, $1 \Leftrightarrow \alpha$, and the same deliberation cost, C. The bounded-rational level in Table 8.2 can be interpreted as a function of the computational power of an agent's deliberation procedure, or a function of the difficulty of an agent's valuation problem. For example, moving from level [1] to level [4], either the valuation problem gets easier, or the agents have more computational resources.

Bounded-rational Level	1 - deliberation effectiveness	Deliberation cost	Initial Deliberation bounds	
	α	C	$\underline{d}$	$\overline{d}$
[1]	0.7	0.5	5	5
[2]	0.3	0.5	4.2	5.8
[3]	0.7	0.05	2.9	7.1
[4]	0.3	0.05	1.8	8.2

Table 8.2: Bounded-rational levels and corresponding initial deliberation bounds

I provide the agents with optimal deliberation and bidding strategies and simulate agent deliberation procedures. In every deliberation step, the new bounds on value are computed such that: (1) the true value remains between the bounds; (2) the true value is uniformly distributed between the bounds with respect to all stochastic sequences of deliberations.

The ask-price in [PP] is selected to maximize expected revenue to the auctioneer, and the minimum bid-increment in [AP] is selected to make sure that agents have positive utility from participation despite the myopia of their metadeliberation strategies.

Second-price Sealed-bid Auction [SB]

We choose the number of deliberations performed by each agent in the [SB] auction off-line, selecting the maximum number of deliberations for which the agents still gain utility through participation in the auction.[3] Figure 8.4 plots efficiency as the number of agents increases, for each level of bounded-rationality, and also the optimal performance (with agents that know the value of the item). The results are averaged over 500 trials.

The sealed-bid auction performs well for small numbers of agents and easy valuation problems (or agents that have sufficient computational-resources), i.e. for up to 20 agents

[3]The agents each perform the following numbers of deliberations for each level of bounded-rationality: $[1, 1, 0, 0, 0, 0, 0, 0, 0, 0]$; $[1, 1, 0, 0, 0, 0, 0, 0, 0, 0]$; $[17, 10, 7, 3, 2, 1, 1, 1, 0, 0]$; $[16, 10, 6, 2, 1, 0, 0, 0, 0, 0]$, for $|\mathcal{I}| = [3, 4, 5, 10, 20, 30, 40, 50, 60, 100]$.

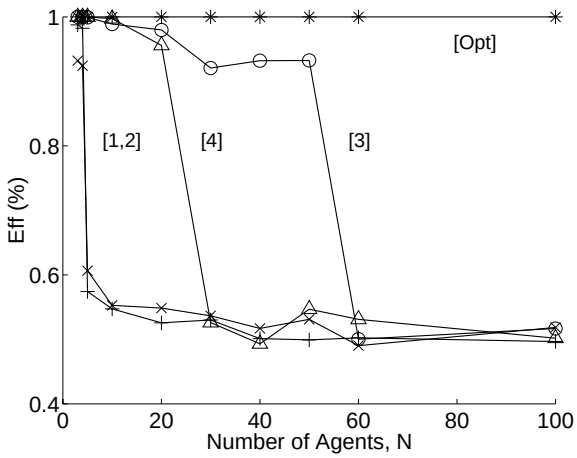


Figure 8.4: Efficiency in the sealed-bid [SB] auction, for agents with bounded-rational levels [1] to [4], and for agents that know their value for the item [Opt].

for type [3] and [4] agents, and for all types of agents when there are less than 5 agents. However, the auction fails at high levels of bounded-rationality, (e.g. [1] and [2]), even with small numbers of agents. No agent can deliberate when the total cost for every agent to perform a single deliberation is greater than the expected utility to the agent that receives the item. I computed pure strategy Nash equilibrium for [SB], in which each agent performs the same number of deliberations.[4] When no agent deliberates the efficiency is approximately 50% because every agent bids $b_i = 5$, and the item is allocated at random. The sealed-bid auction is not bounded-rational compatible, so agents cannot reason effectively about when to deliberate.

It is interesting to compare this analysis with the result that is expected from traditional auction theory, with agents that know their value for the item, as illustrated in line [Opt] in Figure 8.4. The [SB] auction achieves 100% efficiency for all numbers of agents if agents have perfect information about their values, because an agent's dominant strategy is to bid its true value.

[4]In an alternative model, we could compute a mixed-strategy equilibrium for agent deliberation, where agents deliberate m times with probability p. The problem is that there are many such equilibrium, for example 10 agents can deliberate twice for the same cost as 20 agents that deliberate once. There is no simple mechanism for equilibrium selection in a system of uninformed agents.

Sequential Posted-price Auction [PP]

We set the price in the [PP] auction to maximize revenue.[5] Figure 8.5 compares the performance of [PP] 'o' with [SB] 'x' for each bounded-rationality level, as the number of agents increases. Subplot [4] also shows, line '*', the performance of [PP] with agents that know their value for item. The results are averaged over 1000 trials.

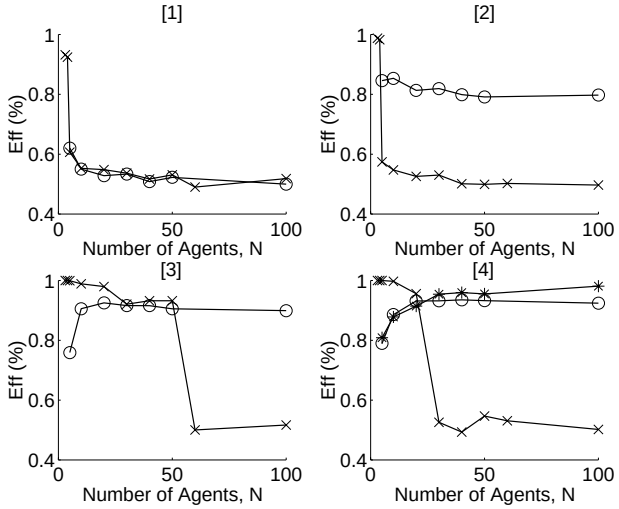


Figure 8.5: Efficiency in the posted-price [PP] 'o' and sealed-bid [SB] 'x' auctions, for agents with bounded-rational levels [1] to [4]. Sub-plot [4] also shows the performance of [PP] with agents that know their value for item (line '$*$').

The [PP] auction achieves low efficiency with small numbers of agents, but tends to support solutions with reasonable efficiency as the number of agents increases. In comparison [SB] is almost 100% efficient for small numbers of agents, but fails for larger numbers of agents. The [PP] auction does not perform well for small numbers of agents because the optimal price varies considerably across trials. Similarly, [PP] with agents that know their value of the item performs worse than [SB] with bounded-rational agents for small numbers of agents. However, when there are many agents, offering agents a fixed price sequentially allows agents to deliberate about their value for the item without losing utility. Agents receive an exclusive offer of the item, at a certain price. There is much more uncertainty in the [SB] auction.

[5]The price is set as follows for each level of bounded-rationality: $p^* = [4.9, 4.9, 4.9,\ 4.9, 4.9, 4.9, 4.9]$, $p^* = [5.8, 5.8, 5.8,\ 5.8, 5.8, 5.8, 5.8]$, $p^* = [6.6, 6.6, 7.0,\ 7.1, 7.1, 7.1, 7.1]$, $p^* = [6.8, 7.8, 8.2,\ 8.2, 8.2, 8.2, 8.2]$, for $|\mathcal{I}| = [5, 10, 20, 30, 40, 50, 100]$. The optimal ask price for agents that know the value of the item is: $p^* = [6.8, 7.8, 8.6,\ 8.8, 9, 9.3, 9.5]$.

The [SB] and [PP] auctions have the same performance with agents that have limited computational resources (level [1]). In this case there is no ask price that will motivate agents to deliberate (see deliberation bounds in Table 8.2), and the best the auctioneer can do is sell the item to the first agent for $p = 4.9$.

Also, notice that with agents that know their value for the item, the efficiency in [PP] approaches 100% as the number of agents increases. In comparison, the performance of [PP] with bounded-rational agents does not approach 100%, even as the number of agents increases. Let us consider type [4] agents. Table 8.2 shows that these agents have initial deliberation bounds $[\underline{d}, \overline{d}] = [1.8, 8.2]$. This means that if the auctioneer charges a price $p > 8.2$, no agent will deliberate, and the item will not be sold because every agent will `reject` the price. The ask price is limited to $p = 8.2$, even for many agents, which limits the ability of the auctioneer to sell the item to the agent with the highest value.

Ascending-price Auction [AP]

The bid-increment in the [AP] auction is set to the minimum value that allows agents to gain positive utility from participation in the auction.[6] Figure 8.6 compares the performance of [AP] '+' with [PP] 'o' for each bounded-rationality level, as the number of agents increases. The results are averaged over 200 trials.

The [AP] auction has very good performance, matching [SB] for small numbers of agents, and matching [PP] for larger numbers of agents. In particular, for agents that have moderate computational resources (i.e. levels [3] and [4]), the [AP] auction generates higher final ask prices than [PP] *and* supports better efficiency. The performance of [AP] holds up as the number of agents increases because agents deliberate in sequence, and wait for other agents to deliberate. Around the same number of agents deliberate for all large numbers of agents, because the bid increment is finite. [PP] performs better than [AP] with agents of type [2], because in this special case the ability to set an exact ask price before the auction is critical to provide incentives for the right agents to deliberate.

[AP] always outperforms [SB], achieving as good a performance when [SB] does not fail, but performing better than [PP] when [SB] fails. This is an interesting result, given the equivalence of the English and Vickrey auctions in traditional auction theory [PMM87].

[6] The bid-increment is set as follows for each level of bounded-rationality: $\epsilon = [1, 1, 1, 1, 1, 1, 1]$, $\epsilon = [0.7,\ 1, 1, 1, 1, 1, 1]$, $\epsilon = [0.2, 0.2, 0.4, 0.4, 0.5, 0.5, 0.5]$, $\epsilon = [0.2, 0.2, 0.6, 0.6, 0.6, 0.6, 0.6]$, for $|\mathcal{I}| = [5, 10, 20, 30, 40, 50,\ 100]$.

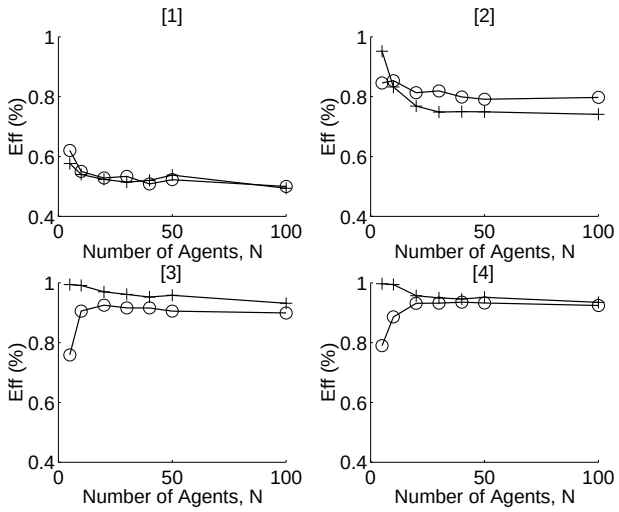


Figure 8.6: Efficiency in the ascending-price [AP] '+' and posted-price [PP] 'o' auctions, for agents with bounded-rational levels [1] to [4].

It is perhaps surprising that the performance of [AP] can be sustained for large numbers of agents. The finite bid increment limits the total amount of deliberation in the auction, even as the number of agents increases.

Mixture of Agents

Let us consider a market with a *mixture* of agents; some with hard valuation problems ("inexperienced agents"), and some with easy valuation problems ("experienced agents"). We assume a fraction f of inexperienced agents; and a fraction $1 \Leftrightarrow f$ of experienced agents that know their value v_i for the item. Fig 8.7 plots performance for $I = 30$, with $\alpha = 0.3$ and $C = 0.05$.[7]

In this example, the performance of [AP] dominates [SB] and [PP] for all mixtures of agents. With many experienced agents, e.g. $f < 0.4$, the [SB] and [AP] auctions are approximately equivalent, and perform better than the [PP] auction. (This is the traditional auction model). For a medium to large fraction of inexperienced agents, $f > 0.4$, the performance of [SB] falls off, while [PP] approaches [AP]. This illustrates the value of a bounded-rational compatible auction ([AP] vs. [SB]) with agents with limited computation.

[7] In [SB] no agents perform any deliberation, In [AP] we choose bid increments: $\epsilon = [0.2, 0.3, 0.4, 0.4, 0.4, 0.6]$ for $f = [0, 0.2, 0.4, 0.6, 0.8, 1.0]$. In [PP] we choose the following ask-price: $p^* = [8.8, 8.6, 8.2, 8.2, 8.2, 8.2]$.

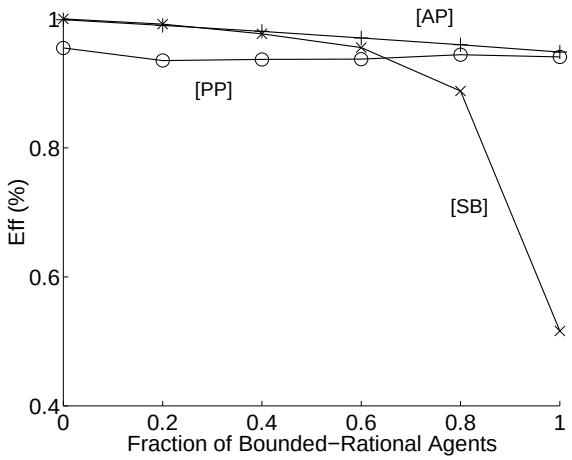


Figure 8.7: Performance of [SB], [PP] and [AP] for a mixture of agent types, with $|\mathcal{I}| = 30$ agents. Fraction f of agents have bounded-rational level [4], while fraction $1 - f$ of agents *know* their value for the item.

Comparing Agent Deliberation across Auctions

In this section we compare agent deliberation in a system with $|\mathcal{I}| = 30$ agents, and $\alpha = 0.7$ and $C = 0.05$ (level [3] in Table 8.2). In this problem the [SB], [AP] and [PP] auctions achieve efficiency of 92.1%, 96.1% and 91.6% respectively. The [AP] auction outperforms the other two auctions. The average utility for an agent that wins each auction is 2.52, 1.56 and 1.87 in the [SB], [AP], and [PP] auctions, so that the auctions can support an average of 1.7, 1.0 and 1.2 deliberations per-agent (dividing by $|\mathcal{I}|C$).

Figure 8.8 (a) plots the average number of deliberations performed by agents in each auction, as a function of the *true* value of the agent for the item, averaged over 500 trials. The agents in [SB] all perform the same number of deliberations, a single deliberation in this case because the utility can support a maximum of 1.7 deliberations per-agent. The average number of deliberations performed in [AP] is 1.0 (compared to a maximum possible of 1.0), and 0.29 deliberations are performed in [PP] (compared to a maximum possible of 1.2).

In fact, 81.2% of agents in [PP] perform no deliberation, including 80.9% of agents with true value between 8 and 10. This occurs when other agents buy the item earlier, and represents a clear loss in allocative efficiency. The seller cannot set a higher price than 7.1, because no agents would deliberate and the item would remain unsold (see the initial upper bound on deliberation in Table 8.2). In comparison, 53.4% of agents in [AP] do not

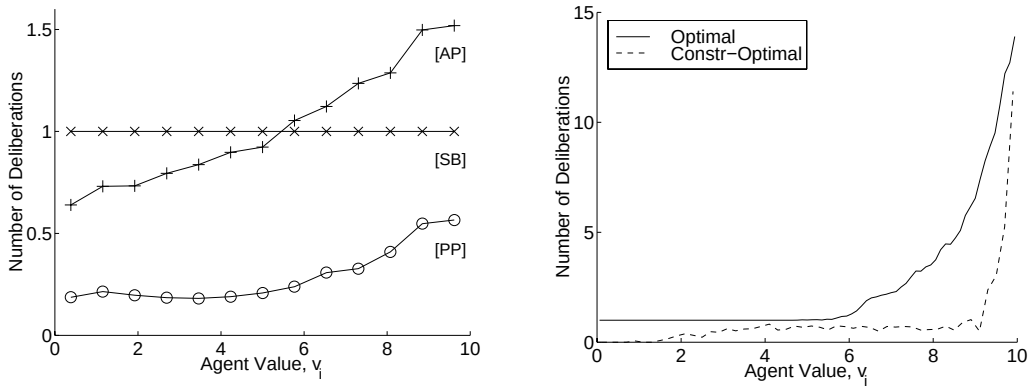


Figure 8.8: Comparison of agent deliberation for $|\mathcal{I}| = 30$ agents with bounded-rational level [4]. (a) Average number of deliberations performed by a single agent, as a function of the agent's (true) value for the item. (b) Average best-case number of deliberations to solve allocation problem (Optimal), and average best-case distribution of a constrained number of total deliberations (1 per agent) to maximize probability of correctness (Constr-Optimal).

deliberate, including 51.8% of agents with value between 8 and 10. In both systems agents with high value for the item choose to deliberate more than agents with low value.

Figure 8.8 (b) plots the best-case deliberation to solve the allocation problem (Optimal), averaged over 500 trials, and the best-case distribution of constrained deliberation (1 per agent) to maximize probability of correctness (Constr-Optimal), averaged over 50 trials.

The *Optimal* deliberation distribution in a problem instance is computed assuming an oracle, that decides which agents should deliberate and cooperative agents. The oracle has perfect information about the true value of each agent, but cannot predict the stochastic nature of the valuation procedure. Deliberation continues until the lower bound on one agent is greater than the upper-bounds on all other agents. The *Constr-Optimal* solution is computed for a total deliberation budget $C_{\max}$ by computing the best solution from the subset of deliberation allocations with increasing deliberation allocated to agents with higher values (this is sufficient).

Notice that the agents with high value deliberate more than the agents with low value in Optimal and in Constr-Optimal. Every agent must deliberate while its upper bound is greater than the lower bound of the agent with the greatest value, and this requires more deliberation for agents with high values. In Optimal the average number of deliberations performed per-agent is 2.6, while the three agents with greatest value perform an average

of 5.8 deliberations each, and the single agent with the greatest value performs an average of 10.2 deliberations.

The [AP] auction allows agents with high values to compute more accurate valuations, and place more accurate bids, than agents with low values. Given a fixed budget for total deliberation (because agents will not lose utility from participation in the auction) this enhances the efficiency of resource-allocations. In [PP] the ask price must be low enough for agents to deliberate, and this can make it impossible to separate agents with high value from agents with medium-high value. Agents with medium-high value might accept the price, while agents with high value are not offered the item. In [AP] the ask-price increases to the agents' initial lower bound on deliberation, and then continues to increase as agents deliberate and bid sequentially. As the price increases it provides incentives for agents with high values to continue to deliberate, while agents with lower values drop out of the auction.

8.5 Limited Computation and Multiple Items

A second set of experiments considered limited computation and multiple items, and looked at the ability to *boost* allocative efficiency with bounded-rational compatible auction design, through the promotion of better metadeliberation. A useful BRC auction will maximize *allocative efficiency* in a particular problem, given self-interested agents with limited computation.

The results highlight the advantages of BRC auctions when agents have limited computation and hard valuation problems, especially in multi-item allocation problems when agents must compute the value of a number of different items or bundles of items. For example, the English auction achieves greater allocative efficiency with less agent computation than a sealed-bid Vickrey auction.

I adopt a simple model of metadeliberation in a decentralized system, the *lazy deliberation and eager bidding* (LDEB) model:

- *Lazy sequential deliberation.* Sequential (*asynchronous*) deliberation ensures that no two agents deliberate at the same time. Agents are selected at random to either bid or deliberate, and deliberate if the value from deliberation is greater than the value from bidding at that moment.

- *Eager bidding.* Agents bid whenever the expected utility from bidding is greater than the expected utility from deliberation, given the current approximate valuation of the agent.

The sequential deliberation model, coupled with *eager bidding* and *instant updates* by the auctioneer of the state of the auction (e.g. ask prices, current allocation) maximizes the value of incremental deliberation by agents. Information about agents' refined values is shared with other agents (via new prices, etc.) before any agent performs additional deliberation.

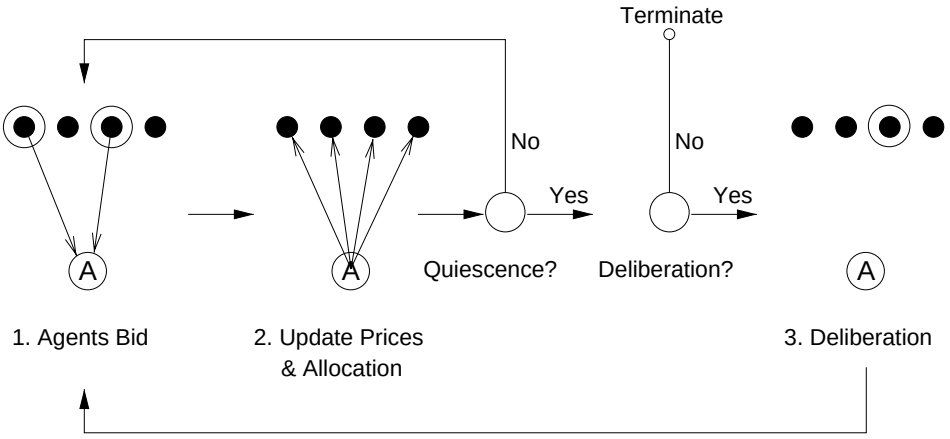


Figure 8.9: The Lazy Deliberation and Eager Bidding agent participation model.

Figure 8.9 illustrates the LDEB model of agent participation. Steps (1), agents with optimal bids place bids with the auctioneer, and (2), the auctioneer updates the state of the auction, repeat until quiescence is reached. Then, (3) either (a) no agent wants to deliberate and the auction terminates, or (b) one agent that prefers to deliberate than have the auction terminate deliberates, and possibly bids. If the agent bids than LDEB returns to step (1), otherwise (3) another agent deliberates or the auction terminates.

The lazy deliberation and eager bidding model provides a general model to compare auction performance with bounded-rational agents. In this case the bid/deliberate decision is determined for agents with *limited* computation, but it could equally well apply to agents with costly computation. The LDEB model is reasonable in a system with fast communication and asynchronous updates, and provides a best-case measure of the allocative efficiency of auctions when agents have limited or costly computation.

8.5.1 Performance Metrics

I propose metrics *bounded-efficiency* and *bounded-computation* to compare the performance of different auctions in a particular problem at design time. The metrics assume a model of agent deliberation procedures, computational resources, and valuation problems.

Bounded-efficiency and bounded-computation are defined as follows:

- Bounded-efficiency $Perf_{\mathcal{A}}(C_{\max})$ of auction $\mathcal{A}$ is the allocative-efficiency achieved with agents that have limited computation budget $C_{\max}$, under model LDEB.
- Bounded-computation $Comp_{\mathcal{A}}(C_{\max})$ of auction $\mathcal{A}$ is the average computation performed by agents with computation budget $C_{\max}$, under model LDEB.

The bounded-efficiency and bounded-computation metrics are computed in simulation with the LDEB model and limited agent computation. The metrics are closely related to the theoretical characterization of BRC auctions. For example, the amount of bounded computation provides a direct measure of the bounded-rational compatibility of an auction:

THEOREM 8.4 *Auction $\mathcal{A}$ is bounded-rational compatible iff the bounded-computation $Comp_{\mathcal{A}}(C_{\text{exact}}) < C_{\text{exact}}$, where C_{exact} is the computation required by an agent to compute exact values for all items.*

In the Vickrey auction $Comp_{\mathcal{A}}(C_{\text{exact}}) = C_{\text{exact}}$, while in the (BRC) English auction $Comp_{\mathcal{A}}(C_{\text{exact}}) < C_{\text{exact}}$ because agents can bid optimally without exact values for items.

An asymptotic measure of the efficiency of an auction mechanism is given by the *maximum* bounded-efficiency of an auction, $Perf^*_{\mathcal{A}}$, computed as:

$$Perf^*_{\mathcal{A}} = \lim_{C \to \infty} Perf_{\mathcal{A}}(C)$$

In the English auction $Perf^*_{\mathcal{A}} = 100\%$, while in a posted-price market $Perf^*_{\mathcal{A}} < 100\%$.

8.5.2 Auction Models

The experiments study three different allocation problems: the single-item problem, the linear-additive problem, and the assignment problem. The linear-additive and assignment problems are defined in the following way:

- *Linear-additive problem.* Allocate $|\mathcal{G}|$ items to maximize total value over all agents, for agents with additive values for items. The value for bundle $S \subseteq \mathcal{G}$ of items $v_i(S) = \sum_{j \in S} v_i(j)$, with items that have values $v_i(j) = U(0, 10)$.

- *Assignment problem.* Allocate $|\mathcal{G}|$ items to maximize total value over all agents, for agents that demand at most one item. The value for bundle $S \subseteq \mathcal{G}$ of items $v_i(S) = \max_{j \in S} v_i(j)$, with items that have values $v_i(j) = U(0, 10)$.

The auction models, i.e. Vickrey, ascending-price and posted-price market, are adapted for each problem domain.

- *Second-price sealed bid auction* (Vickrey). In the additive-value multi-item problem we allow agents to submit OR bids, e.g. (A, p_1) OR (B, p_2) indicates that the agent will pay up to p_1 for item A and up to p_2 for item B, for as many items as desired. To compute the Vickrey outcome separate the bids across items, and run an individual single-item Vickrey auction for each item. In the assignment problem we allow agents to submit XOR bids, e.g. (A, p_1) XOR (B, p_2), indicates that an agent will pay up to p_1 for A *or* up to p_2 for item B but does not want both. The allocation and Vickrey payments in this case can be computed by solving the problem with all agents and then once without each agent in turn. A linear-program will solve the assignment problem.

- *Vickrey-minus.* The Vickrey-minus auction is a Vickrey auction, except in an initial round d agents are eliminated. The dropped agents perform no computation about their values for items and place no bids.

- *Simultaneous Ascending Price Auction.* The simultaneous ascending-price auction implements a separate ascending-price auction for each item. All auctions close simultaneously when every individual auction has reached quiescence.

- *Posted-price.* I consider two simple generalizations of the posted-price market for the additive and assignment multi-item problems. In both cases the auctioneer chooses one price p, the same for all items. In the additive-value problem each item is taken in turn and offered sequentially to agents, with agents selected at random. Items are sold to the first agent to accept the price, and remain unsold if all agents reject the

price. In the assignment problem *all* unsold items, initially all items, are offered to agents, with agents selected at random. Items are sold to the first agent to accept the price, and remain unsold if all agents reject the price.

8.5.3 Agent Metadeliberation

The model of agent computation shares all the features of that in Section 8.4 except that agents now have *limited computation* instead of *costly computation*. Each invocation of the valuation procedure costs an agent a single unit, and each agent has a limited budget $C_{\max}$. An agent selected to deliberate in the LDEB model deliberates if its *expected utility from deliberation* is positive. Otherwise it places its optimal bid, given approximate information about its values for different items.

Agents do not deliberate past a minimal uncertainty on the value of each item, $\Delta_{\min} > 0$, and once the uncertainty is within this minimal error the agent bids with an assumed certain value at the mean of the final bounds. Similarly, when an agent exhausts its deliberation budget it assumes mean values for each item.

In these problems an agent has multiple items to deliberate about. Agents compute independent bounds on the value of each item $j \in \mathcal{G}$. In the additive-value model, this implies that agent i has lower and upper bounds on a bundle $S \subseteq \mathcal{G}$, with lower bound $\underline{v}_i(S) = \sum_{j \in S} \underline{v}_i(j)$ and upper bound $\overline{v}_i(S) = \sum_{j \in S} \overline{v}_i(j)$. In the assignment model, this implies that agent i has lower and upper bounds on a bundle S of $\underline{v}_i(S) = \max_{j \in S} \underline{v}_i(j)$ and $\overline{v}_i(S) = \max_{j \in S} \overline{v}_i(j)$.

The following metadeliberation strategies are implemented for each market and allocation problem:

- *Sealed-bid.* Every agent deliberates until it has accurate values for all items, or until it has used all its computation budget. At each new deliberation step the next item is selected at random.

- *Price-based.* (1) Additive-value problem. Every agent deliberates while the ask price is between its bound on value for one or more items, selecting items at random. The optimal bidding strategy is computed based on upper- and lower- bounds while the deliberation budget is not exhausted, and otherwise based on the mean values. (2) Assignment problem. An agent will bid for an item whenever the utility (value

- price) for one item *dominates* the utility for all other items, i.e. whenever the minimum possible utility from some item j' is at least the maximum possible utility from all other items $j \in G$, $\underline{u}_i(j') = \underline{v}_i(j') \Leftrightarrow p(j') \geq \overline{u}_i(j) = \overline{v}_i(j) \Leftrightarrow p(j)$ for all $j \neq j'$. Otherwise, an agent deliberates about the value of an item that can possibly maximize utility, selecting one of those items at random.

8.5.4 Experimental Results: Limited Computation

The results compare the bounded-efficiency and bounded-computation of each auction in single-item and multi-item allocation problems. In addition to computing bounded-efficiency and bounded-computation, I also compute the average number of times that the final allocation is optimal. This can provide a more sensitive measure of the difference in performance between two auctions than allocative efficiency.

The number of agents are adjusted between 5 and 50, and the number of items fixed at 10. Agents' values for individual items are independently and identically distributed according to a uniform distribution between 0 and 10. The initial approximate value for each agent for an item is $(\underline{v}_i(j), \overline{v}_i(j)) = (0, 10)$, to represent complete uncertainty.

The deliberation parameter is $\alpha = 0.7$, so that an agent reduces uncertainty in the value of an item by 0.3Δ with a single deliberation step, where Δ is the difference between an agent's upper and lower bound for the value of a particular item. The minimum uncertainty $\Delta_{\min} = 0.5$, and agents assume the approximation is accurate when $\overline{v}_i(j) \Leftrightarrow \underline{v}_i(j) < 0.5$ and take the value equal to the mean of the final bounds. Given this, the number of deliberation steps required to compute an exact value for a single item is $C_{\text{exact}} = 9$.

The bid increment in the ascending-price auction is $\epsilon = 0.1$, and the Vickrey-minus auction with between 10–50% dropped agents. The ask price in the posted-price market is set to maximize bounded-efficiency for a computation budget $C_{\max} = 3$, which is 33% of C_{exact}.

The posted-price market requires that the auctioneer has distributional information about agents' values, to set an ask price which maximizes performance on average. In simulation the ask-price is set to maximize bounded-efficiency for a particular agent computation budget, with the same ask price used in all problem instances. The intention is to provide a *best-case* analysis of the performance of a posted-price market. One can consider an auctioneer that has participated in a marketplace for an extended period and has been

able to adjust its ask price to maximize performance.

Single-item Problem

Figures 8.10 (a) and (b) plot the bounded-efficiency and bounded-computation for each auction in the single-item allocation problem, with 20 agents. The results are averaged over 100 trials. In the posted-price market $p^* = 7.3$, and the Vickrey-minus results are presented for 10 (i.e. 50%) eliminated agents. Figure 8.10 (c) plots bounded-efficiency versus bounded-computation, to compare the bounded-efficiency against the computation actually performed by agents.

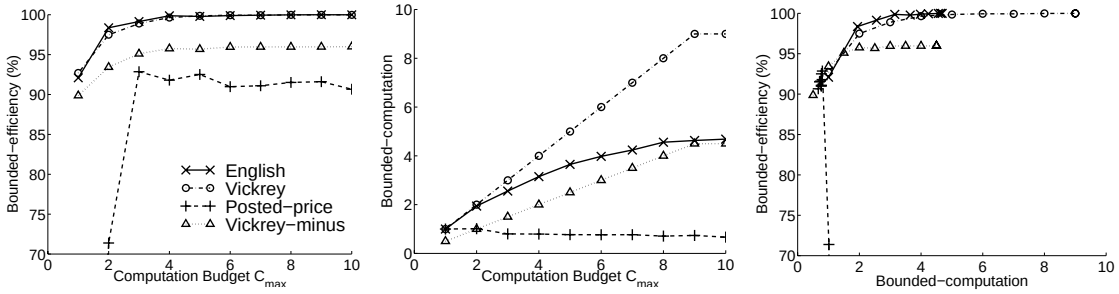


Figure 8.10: Single-item problem with 20 agents. (a) Bounded-efficiency as comp budget increases, (b) Bounded-computation as comp budget increases, (c) Bounded-efficiency vs. bounded-computation.

First, note that the bounded-efficiency in the English, Vickrey and Vickrey-minus auctions increases as $C_{\max}$ increases, because agents perform more deliberation and place more accurate bids. Paradoxically, the performance of the posted-price market actually *falls* slightly for agents with large computation budgets because the price is set to maximize performance for $C_{\max} = 3$. The agents in posted-price perform less average computation as the available computation increases (see Figure 8.10 b). As $C_{\max}$ increases the item is more likely to be sold to an earlier agent, because the agents can compute more accurate values. The same effect is observed for posted-price in the additive-value and assignment problems, Figure 8.11 (a) and Figure 8.12 (a).

In this single item problem the English auction does not outperform the Vickrey with agents that have the same computation budget, Figure 8.10 (a). An agent in the English auction cannot use information about prices to make good allocation decisions; the only decision that an agent can take is to avoid computation, because it only has a single

item for which to compute value. However, the English auction is effective in reducing unnecessary agent computation, achieving 100% bounded-efficiency with 49% less agent computation than the Vickrey auction. This is illustrated in Figures 8.10 (b) and (c).

The posted-price market achieves around 90% bounded-efficiency with as little as 8% of the computation of the Vickrey auction, clearly demonstrating its bounded-rational compatibility. Although the bounded-efficiency of the Vickrey-minus auction is less than for the Vickrey auction, it allows agents to perform 50% less computation on average (Figure 8.10 b), because 50% of agents are eliminated.

I tested the performance of the auctions for different numbers of agents, and found as expected that the performance of the posted-price market and the Vickrey-minus auctions increase with more agents.

Additive-value Problem

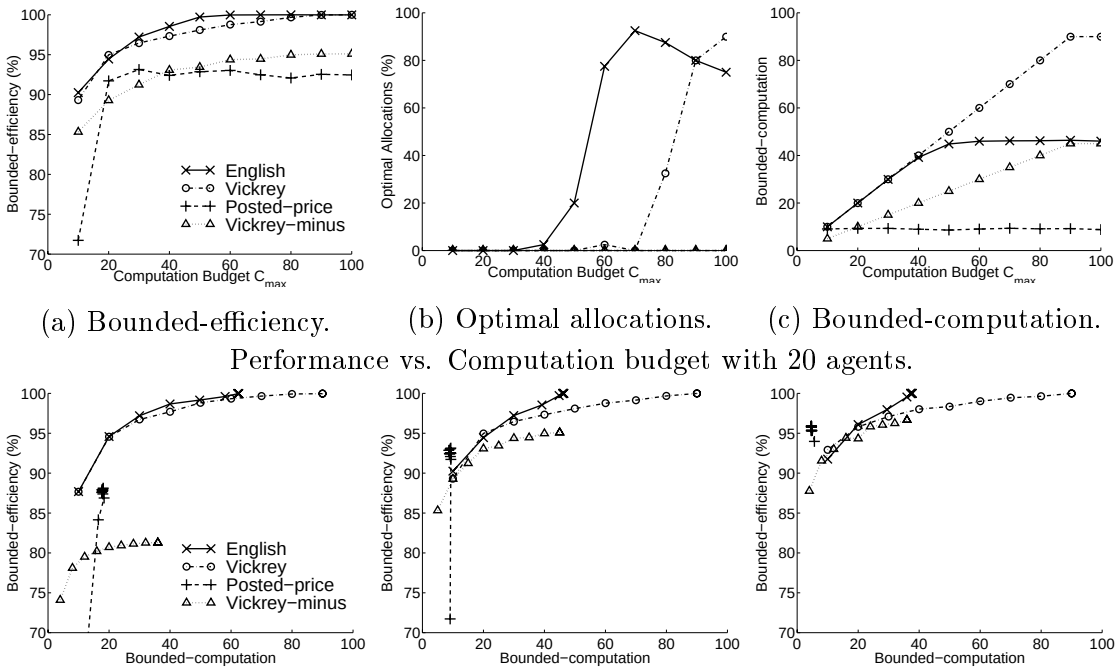
Figure 8.11 plots experimental results for the additive-value multi-item problem with 10 items. The performance of each auction is compared for 5, 20 and 50 agents, computed over 80, 30 and 20 trials respectively. I set the posted price $p^* = 5.6, 8.4$ and 8.8 for each problem size, and drop 50% of agents in Vickrey-minus.

Figures 8.11 (a – c) plot the bounded-efficiency, number of optimal allocations, and bounded-computation for the problem with 20 agents. Figures 8.11 (d – f) plot bounded-efficiency versus bounded-computation for 5, 20 and 50 agents.

In this multi-item problem the English auction performs better than the Vickrey auction for agents with the same computation budget, as shown for example in Figure 8.11 (a) and (b). With 20 agents, for medium budgets, $30 \leq C_{\max} \leq 80$, the bounded-efficiency is greater in the English auction, and the auction computes more optimal allocations. For intermediate computation budgets the agents in the English auction can use prices to make good decisions about how to allocate computational resources.

Furthermore, the agents in the English auction compute 100% efficient allocations with 49% less computation than the agents in the Vickrey auction. This is illustrated in Figure 8.11 (c) and (e). For large computation budgets the agents in the English auction can avoid computation altogether.

Figures 8.11 (d – f) show that as the number of agents increases from 5 to 50, the agents in the English auction are able to avoid more computation on average, computing



(a) Bounded-efficiency. (b) Optimal allocations. (c) Bounded-computation.

Performance vs. Computation budget with 20 agents.

(d) 5 agents. (e) 20 agents. (f) 50 agents.

Bounded-efficiency vs. Bounded-computation.

Figure 8.11: Additive-value problem.

100% efficient allocations are computed with 31%, 49% and 58% less agent computation than in the Vickrey auction.

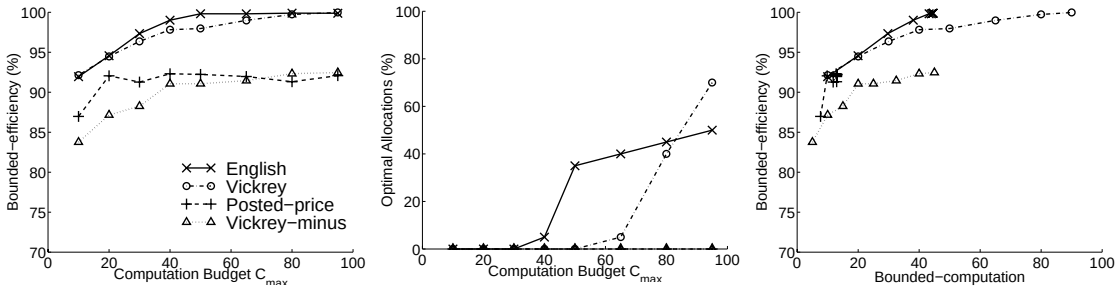
Figures 8.11 (d – f) also show that the posted-price market performs especially well as the number of agents increases, achieving bounded-efficiency of 88%, 93% and 96% for 5, 20, and 50 agents, and with 20%, 10% and 5% of the computation in the Vickrey auction.

The Vickrey-minus auction eliminates 50% of the agents from the auction, and reduces the average computation by 50%. For large numbers of agents it performs almost as well as the Vickrey auction, with 97% bounded-efficiency for 50 agents.

The number of optimal allocations does not reach 100% in either the Vickrey or English auctions, see Figure 8.11 (b). This is because the agents do not refine their value for items beyond uncertainty. This level of accuracy is sufficient for allocations which are approximately 100% efficient, but not sufficient for 100% optimal allocations. We see the same effect for the assignment problem in Figure 8.12 (b).

Assignment Problem

Figure 8.12 plots experimental results for the assignment problem with 20 agents. The results are averaged over 20 trials, with posted-price is $p^* = 6.5$ and 10 dropped agents in Vickrey-minus.



(a) (b) (c)

Figure 8.12: Assignment problem with 20 agents. (a) Bounded-efficiency; (b) Optimal allocations; (c) Bounded-efficiency vs. bounded-computation.

The results are similar to in the additive-value allocation problem. The English auction computes 100% efficient allocations with 51% less computation than the Vickrey auction, and has greater bounded-efficiency for medium computation budgets because agents can use prices to avoid computation on values for items with optimal bids, and perform computation on the value of other items.

The Vickrey-minus auction has smaller bounded-efficiency in this problem than in the additive-value problem. Good allocations require careful coordination between agents in the assignment problem, and dropping just one agent in the optimal allocation can completely change the optimal allocation with the remaining agents. In comparison, good allocations in the additive-value problem are more stable to removing agents.

The posted-price market performs well in this problem. Its bounded-efficiency is 98%, 89% and 92% for 5, 10 and 20 agents, with 40%, 28% and 14% of the agent computation in the Vickrey auction. This suggests that prices which support the optimal allocation tend to be similar for all items.

8.5.5 Discussion

The experimental results demonstrate that the BRC auctions (ascending-price and posted-price) can compute better allocations than non BRC auctions (the Vickrey auction) when

agents have limited computation budgets and must compute the values for multiple items.

In the additive-value and assignment problems the ascending-price auction has greater bounded-efficiency than the Vickrey auction for intermediate computation budgets, and computes optimal allocations for significantly less agent computation as agents' computation budgets increase. Agents can prune local computation based on information about the values of other agents. The effect is particularly noticeable as the number of agents increases, because more agents can avoid computation for incremental computation by any one agent. In comparison, uninformed agents in the Vickrey auction can do no better than select items on which to compute value at random. This is suboptimal when an agent has multiple items to value.

Posted-price markets are useful in systems with many agents, at least if the auctioneer is able to adjust the ask price over time to maximize performance. The posted-price market provides a satisficing approach. Prices can support good solutions with very little computation, because items are often sold before they are offered to many agents and prices allow agents to control computation.

The Vickrey-minus auction eliminates a fraction f of agents and reduces computation by the same amount, f, can perform almost as well as the Vickrey auction in problems with many agents and heavy competition between agents for items. The auction can significantly reduce agent computation for only a small reduction in allocative efficiency. Restricting participation might be useful in systems in which a single-round sealed-bid auction is important, perhaps because communication is unreliable or expensive.

8.6 Related Work

In this section I briefly survey releated work, first in resource-bounded reasoning and metadeliberation, and then in auction theory.

8.6.1 Resource-Bounded Reasoning

Good [Goo71] and Simon [Sim76] provide early discussions on the explicit integration of the costs of deliberation within a framework of agent computation. Later, Horvitz [Hor87] introduced the concept of *bounded-optimality*, that an agent's actions maximize expected utility given its bounds on computation, which is developed in [RSP93].

Horvitz [Hor87] and Boddy & Dean [BD89] introduced the idea of anytime algorithms or *flexible computation*, computational procedures that compute answers to hard problems incrementally. Stochastic *performance profiles* can be associated with anytime algorithms, to allow an agent to reason about the expected value of further deliberation given a model of the cost of further deliberation. In special cases they allow tractable models for normative metadeliberation.

Sandholm [San93, San96] has considered the effect of agent bounded-rationality in distributed task allocation problems. Sandholm [San93] implemented a ContractNet [DS88] based system for a distributed task allocation problem, with agents that bid on the basis of marginal values for tasks. The system, Traconet, allowed agents to bid with approximate values and continue to deliberate during the auction.

In subsequent work Sandholm & Lesser [SL96] propose a framework for *leveled commitment* contracts between agents, which allows agents to decommit from a contract. As noted by the authors, this is useful with agents that have approximate values for tasks and continue to refine their beliefs after striking initial contracts. For example, agents can correct early mistakes as they continue to compute values for tasks. The technique allows agents to integrate local deliberation with negotiation between many other agents.

Sandholm & Lesser [SL97] study a coalition formation problem, in which the problem of computing the value of a coalition structure is complex because it requires determining an optimal assignment of tasks to agents in the coalition. However, the authors do not design a mechanism that allows approximate information about coalition values. Instead it is assumed that agents predict value perfectly. There is no attempt to *integrate* the formation and valuation problems.

Sandholm [San96] demonstrates that the strategy-proofness of an auction can break when agents have approximate values for items and options to continue computation or submit bids. An agent can make a better decision about whether or not to perform further computation about the value of an item if it is well informed about the bids from other agents. As discussed in section 8.2.4, this loss in strategy-proofness can help to *increase* allocative-efficiency so long as the auction is *approximation-proof*, such that truth-revelation of approximate information is a dominant strategy for an agent.

Larson & Sandholm [LS01] model agent deliberation in an auction where agents make explicit decisions about whether to deliberate about their own values or the values of other

agents. The authors determine equilibrium agent strategies and show that whether or not agents engage in *strong strategic deliberation*, which is deliberating about the values of other agents, depends on both the model of bounded-rationality and the auction mechanism. Unlike my analysis Larson & Sandholm do not present any allocative-efficiency comparisons across auction designs with bounded-rational agents.

8.6.2 Auction Models With Costly Participation

A number of economic equilibrium models consider costs associated with participation in an auction, for example costs of bid preparation and information acquisition. However, almost all models assume that all participation decisions are made as a *one-shot decision* before an auction starts, and the models cannot capture the important idea that agents may continue to incur costs as an auction proceeds.

Matthews [Mat84] and Lee [Lee85] model auctions in which there is a cost of *information* acquisition, but all agents have the same value for items. In Matthews [Mat84] agents make a continuous decision about information acquisition, while in Lee [Lee85] agents make a single-shot decision about whether or not to pay to become informed. Both authors conclude that a seller that cares about revenue-maximization might want to *restrict participation*, because it carries the total cost of agent participation. Similarly, Samuelson [Sam85] shows that it can be useful to restrict participation when agents have participation costs (for example bid preparation costs) and individual values for resources. The results are consistent with my work on models of agent bounded-rationality within auctions, if one views the cost of valuations as an external cost of participation.

Kolstad & Guzman [KG99] compute a rational expectations equilibrium for costly information acquisition in a first-price sealed-bid auction, in which agents choose the amount of information to acquire within a cost-benefit model. The information allows agents to tender accurate bids in a construction project. Levin & Smith [LS94] compute mixed-strategy rational-expectations entry strategies in first-price and second-price sealed-bid auctions, for agents with costs of participation that make simultaneous entry decisions based on initial estimates of value. The auctioneer can improve revenue by restricting participation because agents adjust their bids to allow for costs of participation.

Stegeman [Ste96] derives a similar result for a *private-value* auction and agents with one-time participation costs. Neither Levin & Smith or Stegeman can distinguish between

iterative and single-stage auctions because agents decide whether or not to enter the auction *before* the auction starts.

In one of the few models to allow agents to enter sequentially, Ehrman & Peters [EP94] compare the performance of different auctions for agents with one-shot participation costs. The authors show that a *sequential posted-price auction* is useful for high costs of participation because it controls participation, through controlling the number of agents that are offered the item.

Similarly, in a model of *affiliated* values, in which the value of one agent for an item is partially related to the value of other agents, Milgrom & Weber [MW82] show that the English auction outperforms other auctions. The information during the auction, from agents' bids and decisions to leave the auction, allow an agent that remains in the auction to refine its estimate of value. Bids from other agents directly improve an agent's valuation. In comparison, with hard valuation problems and an iterative approximation algorithm bids from other agents provide information that improve an agent's metadeliberation. Milgrom & Weber also show that providing expert appraisals always improves performance (the "linkage principle"), which is analogous to providing free computational resources in a model of bounded-rational agents and hard valuation.

Appendix

In this Appendix I derive the expected utility of deliberation for an agent in the posted-price auction [PP], and present a method for computing the deliberation bounds (Definition 8.10) which are used to determine when an agent should deliberate. Let $\hat{v}_i$ denote the current expected value (mean of bounds).

The expected utility of deliberation, $\hat{u}_i(D^m)$, for a sequence of m deliberations, is computed as the *estimated* increase in utility minus the cost of deliberation:

$$\hat{u_i}(D^m) = \hat{u}_i(b^*_{D^m}) \Leftrightarrow \hat{u}_i(b^*) \Leftrightarrow mC$$

where $\hat{u}_i(b^*)$ is the expected utility of the current optimal bid b^*, and $\hat{u}_i(b^*_{D^m})$ is the expected utility of the optimal bid $b^*_{D^m}$ after deliberation. Both expected values are computed with respect to the agent's belief about its expected value for the item after deliberation.[8]

[8]This is necessary to ensure that the utility of deliberation (before allowing for cost) is always positive. Although deliberation might determine that the current optimal bid has less utility than the agent believed before deliberation, the *true* utility is unchanged.

An agent will deliberate when $\hat{u}_i(D_m) > 0$ for some $m > 0$.

We compute $\hat{u}_i(D^m)$ by case analysis on the ask price:

- $[\overline{v}_i < p]$ *No deliberation.* $\hat{u}_i(D^m) = \Leftrightarrow mC$ for all $m > 0$ because the agent's optimal bid is unchanged (`reject`), and $\hat{u}_i(b^*_{D^m}) = \hat{u}_i(b^*) = 0$.
- $[\underline{v}_i < p < \overline{v}_i]$ *See below.*
- $[p < \underline{v}_i]$ *No deliberation.* $\hat{u}_i(D^m) = \Leftrightarrow mC$ for all $m > 0$ because the agent's optimal bid is unchanged (`accept`), and $\hat{u}_i(b^*_{D^m}) = \hat{u}_i(b^*)$.

The case $\underline{v}_i < p < \overline{v}_i$ can be divided into two subcases, $\hat{v}_i < p < \overline{v_i}$ and $\underline{v_i} < p < \hat{v}_i$. We solve subcase $\hat{v}_i < p < \overline{v_i}$. The solution is valid for the other subcase by symmetry. The utility of deliberation depends on the distance between the ask price p and $\hat{v}_i$. Introduce parameter, γ_i, for the distance between p and $\hat{v}_i$:

$$\gamma_i = \frac{2}{\Delta_i}|p \Leftrightarrow \hat{v}_i|$$

Parameter γ_i is always between 0 and 1, and $\gamma_i = 0$ when $p = \hat{v}_i$, and $\gamma_i = 1$ when $p = \overline{v}_i$, or $p = \underline{v}_i$. In this case the current optimal bid (without deliberation) is `reject`, and $\hat{u}_i(D^m)$ can only be positive if the optimal bid after deliberation is `accept`, that is when $\hat{v}_i > p$ after deliberation. After a sequence of m deliberations the expected value is distributed uniformly: $\hat{v} = U(\underline{v_i} + \alpha^m\Delta/2, \overline{v_i} \Leftrightarrow \alpha^m\Delta/2)$, so the upper bound must be at least p:

$$\overline{v}_i \Leftrightarrow \alpha^m\Delta_i/2 > p$$

Let $m^*(\alpha, C)$ denote the *minimum number of deliberations that can possibly have positive utility*, computed (after substitution for γ) as:

$$m^*(\alpha, \gamma) = \left\lceil \frac{\log(1 \Leftrightarrow \gamma)}{\log(\alpha)} \right\rceil$$

The estimated utility of an agent's optimal bid after $m \geq m^*(\alpha, \gamma)$ deliberations is:

$$\begin{aligned}
\hat{u}_i(b^*_{D^m}) &= \int_{\hat{v}_i = p}^{\overline{v}_i - \alpha^m\Delta/2} \frac{(\hat{v}_i \Leftrightarrow p)}{\Delta(1 \Leftrightarrow \alpha^m)} \mathrm{d}\hat{v}_i \\
&= \frac{(\overline{v}_i \Leftrightarrow \alpha^m\Delta_i/2 \Leftrightarrow p)^2}{2\Delta_i(1 \Leftrightarrow \alpha^m)} \\
&= \frac{\Delta_i(1 \Leftrightarrow \gamma \Leftrightarrow \alpha^m)}{8(1 \Leftrightarrow \alpha^m)}
\end{aligned}$$

because $\hat{v}_i = U(\underline{v}_i + \alpha^m \Delta_i/2, \overline{v}_i \Leftrightarrow \alpha^m \Delta_i/2)$, and the agent will `accept` the price if $\hat{v}_i > p$, for expected utility $\hat{v}_i \Leftrightarrow p$.

Putting everything together, because $\hat{u}_i(b^*) = 0$ (the current optimal bid is `reject`), the expected utility of deliberation in this case is:

$$\hat{u}_i(D^m) = \begin{cases} \frac{\Delta_i(1-\gamma-\alpha^m)^2}{8(1-\alpha^m)} \Leftrightarrow mC & \text{, if } m \geq m^*(\alpha,\gamma) \\ \Leftrightarrow mC & \text{, otherwise.} \end{cases}$$

Comparative statics confirms our intuition about the value of deliberation to an agent. Deliberation is more useful as: the price gets closer to an agent's expected value (because it is more likely to change an agent's bid); the uncertainty increases; the deliberation effectiveness increases; and the deliberation cost decreases.

- $\partial \hat{u}_i(D^m)/\partial \gamma < 0$

  Proof. By contradiction. Assume $\partial \hat{u}_i(D^m)/\partial \gamma = \Leftrightarrow 2(1 \Leftrightarrow \gamma \Leftrightarrow \alpha^m)/8(1 \Leftrightarrow \alpha^m)$ is positive, for some $m \geq m^*(\alpha,\gamma)$. This implies that $1 \Leftrightarrow \gamma < \alpha^m$. However, $m \geq m^*(\alpha,\gamma)$ iff $1 \Leftrightarrow \gamma > \alpha^m$, giving a contradiction ∎

- $\partial \hat{u}_i(D^m)/\partial \Delta > 0$, when $p = \hat{v}_i$

  Proof. With $\gamma = 0$, we have $\hat{u}_i(D^m) = \Delta_i(1 \Leftrightarrow \alpha^m)/8$, and $\partial \hat{u}_i(D^m)/\partial \Delta_i = (1 \Leftrightarrow \alpha^m)/8$, positive because $\alpha^m < 1$ ∎

- $\partial \hat{u}_i(D^m)/\partial \alpha < 0$, when $p = \hat{v}_i$

  Proof. With $\gamma = 0$, $\partial \hat{u}_i(D^m)/\partial \alpha = \Leftrightarrow m\Delta\alpha^{m-1}/8$, which is negative ∎

- $\partial \hat{u}_i(D^m)/\partial C < 0$

  Proof. Trivial, because $\partial \hat{u}_i(D^m)/\partial C = \Leftrightarrow m$ ∎

We can compute the *deliberation bounds* as the largest γ for which $\hat{u}_i(D^m) > 0$ for some $m > 0$. When no such γ exists, then $\overline{d}_i = \underline{d}_i = \hat{v}_i$ and the agent will not deliberate at all, for any ask price.

Chapter 9

Extended Example: Distributed Train Scheduling

In this chapter I present a computational study of an auction-based method for decentralized train scheduling.[1] Auction methods are well suited to the natural information and control structure of modern railroads. I assume separate network territories, with an autonomous dispatch agent responsible for the flow of trains over each territory. Each train is represented by a self-interested agent that bids for the right to travel across the network from its source to destination, submitting bids to multiple dispatch agents along its route as necessary.

The natural separation of track control across multiple dispatch agents precludes a combinatorial auction for the entire problem. Instead each individual dispatcher runs an auction for the right to travel across its territory, and trains must solve a coordination problem to receive compatible entry and exit times across their complete route. Trains bid for the right to enter and exit a territory at particular times. The dispatcher agents selects bids to maximize revenue in each round. Feasibility requires that there is a safe schedule for trains over the dispatcher's region given the bid times. As such, the scheduling problem lies outside of the standard combinatorial allocation problem.

An additional difficulty in this domain is presented by the continuous time dimension to a bid. Instead of imposing a finite time discretization on the system I provide an expressive bidding language that allows trains to bid for the right to enter and exit within *ranges* of times (e.g. "arriving no later than 12 pm"). Prices are approximated with a finite grid of (entry, exit) pairs, and updated by dispatcher agents with *i*Bundle-style price update rules.

[1] This chapter draws a lot of material from Parkes & Ungar [PU01].

Computational results on a simple network with straight-forward best-response bidding strategies demonstrate that the auction computes near-optimal system-wide schedules. In addition, the method appears to have useful scaling properties, both with the number of trains and with the number of dispatchers, and generates less extremal solutions than those obtained using traditional centralized optimization techniques.

9.1 Introduction

Auction-based scheduling methods are well-suited to the decentralized information and control structure of modern railroads. The flow of trains over a railroad network is not controlled by a single centralized scheduler, but rather by the joint decisions of a number of largely autonomous dispatcher agents, each responsible for a local track territory. In addition, trains are operated by competing companies, each of which would prefer for their trains to run on-schedule even if the trains of other companies must wait. Real train drivers receive bonuses for on-line arrivals, and have private information about repair schedules, etc.

Auction-based methods fill two important needs. First, they respect the natural autonomy and private information within such a distributed system. Secondly, they can provide incentives for trains to reveal truthful information (indirectly, via bids) about their values for different schedules. In a naive central implementation, a self-interested train with private information about its time constraints, value, and costs, cannot be expected to act truthfully, but rather to misrepresent this information if it will improve its own schedule in the system-wide solution.

The train scheduling problem that I address in this chapter falls within a hierarchy of interrelated train scheduling problems; see [KH95] for a recent survey. I assume that all *strategic planning*, i.e. deciding on train routes and assigning values, times, and costs, is already completed. The input is a set of trains, each with a defined routes over a track network, a value for completing its journey, and an optimal departure and arrival time and cost function for off-schedule performance. The system-wide problem is to compute a robust and safe schedule for the movement of trains over the network to maximize the total cost-adjusted value over all trains.

In constructing an optimal schedule I depart from earlier models for automatic train

scheduling that used fixed-priority scheduling rules.[2] These early models have been criticized for a "hurry up and wait" approach [KHC91], with high priority trains moved down lines as fast as possible, possibly causing problems and inefficiencies at yards further down the line. I build instead on the *pacing models* of Kraay *et al.* [KHC91], which control the speed of trains in finding optimal schedules.

The auction design has each dispatcher agent running a separate auction, for the right to enter and exit its territory at particular times. There are necessarily *multiple* auctions, to respect the autonomy of individual dispatchers to make local decisions. All auctions terminate simultaneously, when there is quiescence across the system.

A train agent must bid for pairs of entry and exit times across multiple dispatchers to complete its journey, which presents a coordination problem. The exit time from one dispatcher must be early enough to allow the train to enter the next dispatcher on its route at the required entry time. Iterative auctions (as opposed to sealed-bid auctions) allow trains to adjust towards a good solution, and should help to solve this coordination problem. In addition, trains can submit bids for *sets* of times, i.e. "I want to enter your territory at any time after 10 am, but leave no later than 1.30 pm, and my maximum travel speed is 100 km/hr." This constraint-based bidding language is a concise way to handle the continuous time attribute of a bid without imposing an explicit discretization on time.

In each round the auctioneer computes the set of bids that maximize revenue, subject to the constraint that there must be a safe schedule for trains given the entry and exit times in accepted bids. The winner-determination problem is solved without a discretization on time, formulated as a mixed integer program.

Although there is no explicit discretization on time imposed on agents' bids, the *prices* in the auction are maintained over a discrete price lattice. The lattice maintains prices on *pairs* of entry and exit times, and prices are computed on-the-fly for any pair of entry-exit times based on the closest lattice points. The discrete price lattice *does not* restrict the times that can receive bids, but rather provides an approximate method to maintain prices. Prices are increased across rounds with an *i*Bundle style price-update, i.e. based on the bid prices from unsuccessful agents.

[2]Priority-based dispatching is still used in most North American railroads; high priority trains are generally not delayed even if they are running early, while low priority trains are delayed even if they are running late [Hal93].

Experimental results compare the quality of schedules computed in the auction-based method with schedules computed under a traditional centralized optimization approach. In order to make a fair computational comparison across the methods, the global scheduling problem and the winner-determination problems are both formulated as (closely related) mixed integer programs, and solved with CPLEX— a standard mixed integer programming software package. The experiments compare centralized solutions (with complete information about agents' preferences) with auction-based solutions, on a set of stochastic problem instances.

In assessing the performance of the auction-based method I make a reasonable assumption about agent bidding strategies, i.e. that agents follow a *myopic best-response bidding strategy*, and submit bids to maximize value given the current ask prices.

The computational results demonstrate that the auction-based method can generate *better* schedules than the centralized method, and in less time. Moreover, the auction-based method appears to have good scaling properties with the number of agents and dispatchers, at least for the auction parameters selected in the tests (e.g. price update speed, time in each round to solve winner-determination, etc.)

Further experimentation is required to make a full assessment of the auction-based method's computational properties. I expect that the performance of the simple myopic bidding strategy might begin to fall-off as the number of dispatchers continues to increase, as agents' bid coordination problem becomes more difficult. The myopic bidding approach, in which agents bid across all dispatchers simultaneously in response to current prices can leave agents "exposed" to times that they cannot fit with times from other dispatchers. Agents would require alternative more sophisticated bidding strategies in these cases, to avoid this exposure problem.

The outline of the rest of this chapter is as follows. In Section 9.2 I define the *global* train-scheduling problem, and formulate a mixed-integer program model under assumptions of centralized information about the local values and cost functions of each train. In Section 9.3 I describe the key elements of the *auction-based solution*: the bidding language, price-updates, winner-determination, termination conditions, and bidding rules. Section 9.4 formulates the bidding problem for a train agent in the auction as a shortest-path problem, which is solved using dynamic-programming. Finally, Section 9.5 presents experimental results over a set of stochastic train scheduling problems.

9.2 The Train Scheduling Problem

Each train is assumed to have a *source* and *destination* node, a value to complete its journey, and a cost for off-time departure and arrival. The global objective is to find a safe schedule that maximizes the total *net* value, the total value minus cost of delay across all trains that run. I introduce a novel mixed-integer programming (MIP) formulation that allows trains to be dropped when necessary, i.e. to allow other high-valued trains to run on-time. A very similar formulation is adopted for the winner-determination problem in the auction (see the next section).

9.2.1 Track Network: Topology and Constraints

In modeling the train scheduling problem I make a number of simplifying assumptions, both about the network structure and about the types of interactions that are allowed between trains.

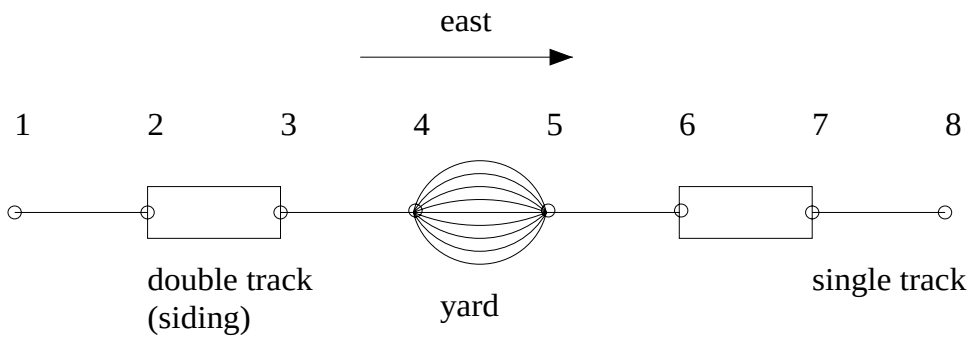


Figure 9.1: The train scheduling problem.

The key assumption is that of a *single line* operation— a sequence of *single-track*, *double-track*, or *yard* sections, separated by *nodes*. The structure is illustrated in Figure 9.1. The single-line operation simplifies the specification of the global train-scheduling problem and the winner-determination problem in the auction-based method. The single line assumption also allows train agents to restrict their attention to tradeoffs across multiple temporally different routes, ignoring alternate paths over the network. The same assumption is made across much of the train scheduling literature, for example in Kraay *et al.* [KHC91], Kraay & Harker [KH95], and Hallowell [Hal93]. Section 9.7 discusses a possible extension of this auction-based method to a multiple line network.

An interaction between a pair of trains may be a *meet* or a *pass*, and is associated with a network location and a time. A meet is when two trains traveling in *opposite directions* are at the same location at the same time. A pass is when two trains traveling in the *same direction* are at the same location at the same time.

The *feasibility* of a schedule for trains across a network is determined by the *safety* of meets and passes. This depends on the type of section:

(**S1**) Any number of trains can meet and pass in yards.

(**S2**) Any number of trains can meet on double-track sections, but no trains can pass.

(**S3**) No trains can meet or pass on a single-track section.

In addition, a feasible schedule must maintain a *minimum separation distance*, Δ_{safety}, between trains on single and double track sections. This minimum separation distance requirement is waived for trains in yards (but not on exit into neighboring sections). Finally, no train can exceed either its maximum speed or the maximum safe speed on any section.

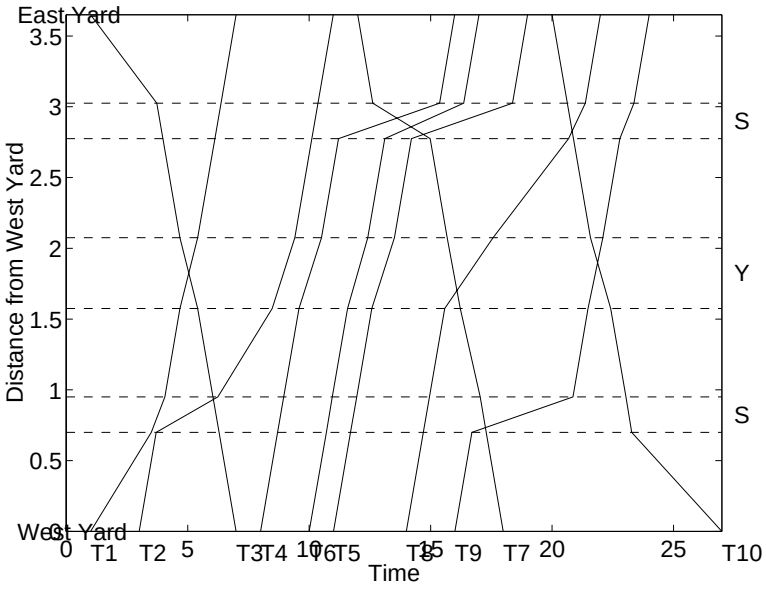


Figure 9.2: A safe train schedule.

Allowing trains to meet but not pass on double-track sections reduces problem-solving complexity because there are many ways for two trains to cross in the same direction but only a few ways for two trains to cross in opposite directions. Similarly, modeling infinite-capacity yards and double-track sections (sidings) is a simplifying assumption.

Figure 9.2 illustrates a safe train schedule for a network with the track topology illustrated in Figure 9.1. The exact parameters in this example are as described in the experimental results in Section 9.5; in this case with ten trains of which trains 1, 2, 4, 5, 6, 8, 9 run East. The sidings (or double-track sections) in this network are illustrated with an "S" on the right of the plot, the yard with a "Y". Notice that trains meet at sidings and at the yard, but no trains meet or pass on the single track sections. In this example no trains pass in the yard, but this is not precluded in the model. Notice also that the trains remain the *safety* distance apart, and are expected to maintain constant speeds within each section.

9.2.2 Schedules

A schedule specifies the network position across time for each train in the system. It is sufficient to consider schedules in which trains travel at a constant speed with each section (the speed can vary from train to train and from section to section), by the following result:

LEMMA 9.1 *Any feasible schedule can be reduced to a feasible schedule where each train travels at a constant speed within each track section.*

The transformation that maintains feasibility is to hold times at nodes between track sections constant, and *smooth* the speed of each train between these points. The proof is quite straightforward— just show that the number of meets are the same for any speed profile consistent with the entry and exit points, and that the number of passes is (weakly) less when trains travel at a constant speed. I choose to ignore constraints on acceleration across sections.

This observation reduces the size of the search space in the scheduling problem, and simplifies the problem of finding optimal times for trains at the ends of each section.

9.2.3 A Mixed Integer Programming Formulation

Let $\mathcal{I}$ denote the set of trains and $\mathcal{N}$ denote the set of nodes between track sections. It is useful to view the network in a west–east orientation, with nodes ordered such that $j > k$ for $j \in \mathcal{N}$ further east than $k \in \mathcal{N}$. The trains are divided into a set *east* $\subseteq \mathcal{I}$ that travel west-to-east and *west* that travel east-to-west. The nodes are labeled with

the type of section to the east, e.g. the section between node j and $j+1$ is a yard if $j \in \mathit{yard}$, single-track if $j \in \mathit{single}$, and double-track otherwise. The minimum travel time for train i between node j and $j+1$, its *free-running time* $r(i,j)$, is defined by the length of the section, the maximum speed of the train, and the maximum safe speed over the section. This is the time to run from west-to-east for a train $i \in \mathit{east}$, and from east-to-west otherwise. Later, when I formulate the MIP for winner-determination I will leave this information implicit in the bidding language to simplify the presentation.

Each train $i \in \mathcal{I}$ has a *source node* and optimal departure time, $(s(i), t_{\mathrm{s}}^*(i))$, a *destination node* and optimal arrival time, $(d(i), t_{\mathrm{d}}^*(i))$, a *value* $V_i \geq 0$ for completing its journey, and a *cost penalty*, $cost_i(t_{\mathrm{s}}, t_{\mathrm{d}})$, for off-schedule performance. Following [Hal93] we assume a linear additive cost penalty for each train. Given actual source t_{s} and destination t_{d} times for train i, the cost for off-schedule performance is computed as:

$$cost_i(t_{\mathrm{s}}, t_{\mathrm{d}}) = C_i \mid t_{\mathrm{s}} \Leftrightarrow t_{\mathrm{s}}^*(i)| + C_i \mid t_{\mathrm{d}} \Leftrightarrow t_{\mathrm{d}}^*(i)|$$

where $C_i > 0$ is train i's *unit cost for off-schedule performance.*

This cost function assumes that performance is measured only on the basis of a train's time at its source and destination nodes. This is reasonable for a freight train with a single shipment to make, but less appropriate for a train that must make intermediate scheduled stops.

Given that it is sufficient to consider only trains that travel at a constant speed across each section (Lemma 9.1), we can specify a schedule with the time, $t(i,j)$, of each train i at node j. A train can be dropped from the schedule. Let $y(i) \in \{0,1\}$ equal 1 if train i is not dropped from the schedule, and 0 otherwise. Let $\Delta_{\mathrm{source}}(i)$ and $\Delta_{\mathrm{dest}}(i)$ denote the *absolute* error in departure and arrival time for train i at source node $s(i)$ and destination node $d(i)$. The system-wide objective is to maximize total value minus cost:

$$\max \sum_i V_i y(i) \Leftrightarrow \sum_i C_i \Delta_{\mathrm{source}}(i) \Leftrightarrow \sum_i C_i \Delta_{\mathrm{dest}}(i)$$

In congested networks with high cost penalties it can be better to drop low value trains in allow more trains to run on schedule and avoid cost penalties. Dropped trains neither achieve any value nor incur any cost penalties. This is achieved by a smart formulation of the method to compute arrival and destination errors, $\Delta_{\mathrm{source}}(i)$ and $\Delta_{\mathrm{dest}}(i)$.

The constraints make sure schedules are feasible, i.e. that a schedule is safe, trains are separated, and speed constraints are not violated. In the following ("the big M technique"),

M is a large positive number, used to make sure that dropped trains do not restrict schedules for other trains, to allow dropped trains to incur zero penalties, and to implement disjunctive logic constraints as a mixed-integer program.

Constraints (1a) and (1b) set the errors $\Delta_{\text{source}}(i)$ and $\Delta_{\text{dest}}(i)$ for train i:

$$\Delta_{\text{source}}(i) \geq |t(i, s(i)) \Leftrightarrow t^*_{\text{s}}(i)| \Leftrightarrow M(1 \Leftrightarrow y(i)) \quad \forall i \in \mathcal{I} \tag{1a}$$

$$\Delta_{\text{dest}}(i) \geq |t(i, d(i)) \Leftrightarrow t^*_{\text{d}}(i)| \Leftrightarrow M(1 \Leftrightarrow y(i)) \quad \forall i \in \mathcal{I} \tag{1b}$$

The absolute value constraint can be implemented by writing two greater than constraints, one for the positive term and one for the negative term.

Notice that if $y(i) = 0$ for train i then $\Delta(i) = 0$ is a solution, and we count no penalty for dropped trains. This avoids requiring non-linear terms, such as $C_i \Delta_{\text{source}}(i) y(i)$, in the objective function.

Constraints (2a) and (2b) ensure consistency of travel times for trains, given free running time $r(i, j)$ for train i between node j and $j + 1$. Again, neither constraint is binding for a dropped train by the "big M" formulation.

$$t(i, j+1) \geq t(i, j) + r(i, j) \Leftrightarrow M(1 \Leftrightarrow y(i)) \quad \forall i \in \mathit{east}, \forall j \in \mathcal{N} \tag{2a}$$

$$t(i, j+1) \leq t(i, j) \Leftrightarrow r(i, j) + M(1 \Leftrightarrow y(i)) \quad \forall i \in \mathit{west}, \forall j \in \mathcal{N} \tag{2b}$$

The zero-one variables $\mathit{gap}(i, i', j)$ make sure that trains are a safe distance apart at all times; $\mathit{gap}(i, i', j) = 1$ iff train i trails train i' by at least time *safety* at node j. The "big M" technique is used to constrain a train to be either more than *safety* ahead *or* more than *safety* behind another train. Note that constraint (3b) is true whenever at least one of the trains is dropped, so that the times on dropped trains are not constrained.

$$t(i, j) \Leftrightarrow t(i', j) + M\mathit{gap}(i, i', j) \geq \mathit{safety} \quad , \forall j \in \mathcal{N}, \forall i, i' \in \mathcal{I} \tag{3a}$$

$$t(i', j) \Leftrightarrow t(i, j) + M(1 \Leftrightarrow \mathit{gap}(i, i', j)) + M(2 \Leftrightarrow y(i) \Leftrightarrow y(i')) \geq \mathit{safety}, \forall j \in \mathcal{N}, \forall i, i' \in \mathcal{I} \tag{3b}$$

The zero-one variables $\mathit{after}(i, i', j)$ indicate whether train i arrives at node j after train i'; $\mathit{after}(i, i', j) = 1$ if train i is after train i' at node j. This indicator variable is set by constraints (4a) and (4b) to be consistent with the times defined by variables $t(i, i', j)$. A dropped train can assume the same ordering with respect to all trains, allowing them

to trivially satisfy (4c) and (4d).

$$t(i,j) \Leftrightarrow t(i',j) \leq M after(i,i',j) \quad ,\forall j \in \mathcal{N}, \forall i,i' \in \mathcal{I} \tag{4a}$$

$$t(i',j) \Leftrightarrow t(i,j) \Leftrightarrow M(2 \Leftrightarrow y(i) \Leftrightarrow y(i')) \leq$$
$$M(1 \Leftrightarrow after(i,i',j)) \quad ,\forall j \in \mathcal{N}, \forall i,i' \in \mathcal{I} \tag{4b}$$

Constraint (4c) captures the restriction that trains traveling in the same direction cannot pass on sidings or single-track sections. East-bound train i must remain after east-bound train i' at node j if it is behind train i at node $j+1$ and the section between j and $j+1$ is not a yard. Similarly for west-bound trains.

$$after(i,i',j) = after(i,i',j+1)$$
$$\forall j \notin yard, \forall i,i' \in east, \forall i,i' \in west \tag{4c}$$

Finally, constraint (4d) captures the restriction that trains traveling in opposite directions cannot meet on single-track sections. If east-bound train i is after west-bound train i' at node j it must also have followed west-bound train i' at node $j+1$ for single-track sections between j and $j+1$, otherwise the trains were on the same single-track section at the same time and traveling in opposite directions.

$$after(i,i',j) = after(i,i',j+1)$$
$$\forall j \in single, \forall i \in east, \forall i' \in west \tag{4d}$$

Taken together with the objective function, the constraints specify a mixed-integer program to solve the centralized train scheduling problem. The optimal solution specifies which trains are dropped (with $y(i) = 0$) and the times $t(i,j)$ for other trains at each node j in the network.

9.3 An Auction-Based Solution

In introducing the auction-based solution, let us assume that the track network is divided across dispatcher territories, with each dispatcher responsible for the local flow of trains. A separate dispatcher agent auctions the right to travel across each territory. Each train is associated with a train agent that places bids for the right to travel across a territory, and coordinates times across dispatchers on its route to achieve a good schedule. The

dispatchers have information about the local network topology, i.e. the location of the double-track network sections and the location of the yards.

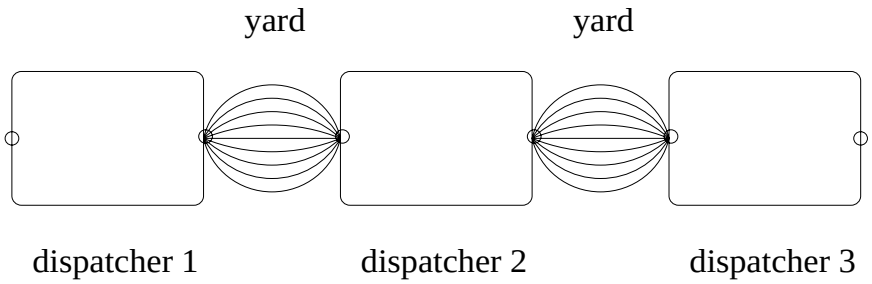


Figure 9.3: The dispatcher territory structure.

The decentralized problem structure is illustrated in Figure 9.3. I assume that dispatch territories are separated by neutral yards. This allows the safety constraints on meets and passes to be decoupled across dispatch territories because yards have infinite capacity and allow arbitrary meets and passes. The dispatcher on each side of a connecting yard must simply ensure that trains remain a *safety* distance apart as they enter and exit its territory.

9.3.1 Auction Innovations

The structure of the train scheduling problem requires a number of innovations in auction design:
(1) A constraint-based bidding language to allow train agents to submit bids with continuous time attributes, and represent a choice set over different pairs of times.
(2) An approximate representation of a continuous and non-linear price space. Prices are maintained over a discrete lattice with quotes computed on-the-fly for any pair of times.
(3) An integer-programming method to compute feasible train schedules given entry and exit times in bids, and check conflicts across bids.

As noted earlier, there are multiple independent auctions, one for each dispatcher territory. This respects the decision autonomy of each dispatcher. Given that trains must receive compatible entry-exit times across multiple dispatchers, the auctions are necessarily iterative to allow train agents to coordinate their bids across multiple auctions. All auctions close simultaneously when bid quiescence is detected across the system.

9.3.2 Dispatcher Auction

Each dispatcher runs an ascending-price auction, maintaining *ask prices* for (departure, arrival) times at pairs of nodes, and a provisional schedule. The ask price is a *lower bound* on an acceptable bid price. A provisional schedule is computed in each round to maximize revenue, based on bids from train agents. Train agents can bid for *entry* and *exit* times in a territory, while the dispatcher agents have the flexibility to decide exactly how a train will run, so long as the schedule is consistent with those times. In the following I describe a single dispatcher auction in some detail.

Bidding Language

The bidding language is quite expressive. A train can bid to enter a territory at time t_{entry} and depart at time t_{exit}, and state whether each time is *fixed* or *flexible*. With a fixed time the train must enter (or exit) the territory at that exact time. With a flexible entry time, any time *after* t_{entry} is acceptable; with a flexible exit time, any time *before* t_{exit} is acceptable (subject to constraints on a train's minimal travel time). Finally, a train agent can submit multiple bids to the same dispatcher, coupled with an "exclusive-or" constraint, to state that the dispatcher can accept any *one* pair of times.

Let $\mathcal{K}$ denote the set of all bids received by a dispatcher, and $\beta(i) \subseteq \mathcal{K}$ the bids received from agent i. A set of bids from agent i in a particular round are all associated with a single entry node, $n_{\text{entry}}(i)$, a single exit node $n_{\text{exit}}(i)$, and true/false values $\mathit{fixed}_{\text{entry}}$ and $\mathit{fixed}_{\text{exit}}$ to state whether the times are fixed or flexible. Each individual bid $k \in \beta(i)$ specifies an entry time, $t_{\text{entry}}(k)$, an exit time $t_{\text{exit}}(k)$, and a bid price $p(k) \geq 0$.

Example:

Bid $(5, 10, \$100)$ xor $(7, 12, \$150)$ for entry node A and exit node B, with $\mathit{fixed}_{\text{entry}}$ but $\neg\mathit{fixed}_{\text{exit}}$, states that the train agent is willing to pay up to \$100 to enter at A at time 5 and depart before time 10, or up to \$150 to enter at time 7 and depart before time 12.

To keep the winner-determination problem tractable I also find it useful to restrict the number of bids that an agent can place in any round.[3]

[3] Experimentally, $B_{\max} = 5$ appears to work well in many problems.

Winner-determination

The auction has multiple rounds. In each round the dispatcher solves the winner-determination problem, computing a provisional allocation to maximize revenue based on bids. The provisional allocation must be consistent with some feasible schedule.

The winner determination problem is formulated as a mixed-integer program that is very similar in form to that for the centralized train scheduling problem. However, it tends to be much easier to solve because the problem is restricted to the space of solutions compatible with the bids submitted by agents and the problem is for only a single territory.

As in the centralized problem, it is not necessary to select a bid from every train agent even if there is a feasible solution involving every agent. Sometimes a schedule with fewer agents will generate more revenue.

Borrowing as much from the earlier global MIP formulation (see Section 9.2.3) as possible, we introduce new zero-one variables $x(i,k) \in \{0,1\}$ for agent $i \in \mathcal{I}$ and bid $k \in \beta(i)$, where $\beta(i)$ is the set of bids from agent i, with $x(i,k) = 1$ iff agent i's bid k is in the provisional allocation. The linear objective function is:

$$\max \sum_{i \in \mathcal{I}} \sum_{k \in \beta(i)} p(k) x(i,k)$$

i.e. accept bids to maximize total revenue where $p(k)$ denotes the bid price of bid k from agent i.

Constraints (2a, 2b, 3a, 3b, 4a, 4b, 4c, 4d) are retained from the MIP of the global train scheduling problem, with train times computed on the basis of bids from agents. Allowing for flexible bid times, we write:

$$t(i,s(i)) = \sum_{k \in \beta(i)} t_{\text{entry}}(k) x(i,k) \quad \text{, if } \mathit{fixed}_{\text{entry}}(i) \tag{1a'}$$

$$t(i,s(i)) \geq \sum_{k \in \beta(i)} t_{\text{entry}}(k) x(i,k) \quad \text{, otherwise}$$

$$t(i,d(i)) = \sum_{k \in \beta(i)} t_{\text{exit}}(k) x(i,k) \quad \text{, if } \mathit{fixed}_{\text{exit}}(i) \tag{1b'}$$

$$t(i,d(i)) \leq \sum_{k \in \beta(i)} t_{\text{exit}}(k) x(i,k) \quad \text{, otherwise}$$

$$\sum_{k \in \beta(i)} x(i,k) \leq y(i) \tag{1c'}$$

for every train $i \in \mathcal{I}$.

The source node $s(i)$ is the entry node $n_{\text{entry}}(i)$, and the destination node $d(i)$ is the exit node, $n_{\text{exit}}(i)$. Constraints (1a') constrain the schedule for train i to an entry time consistent with its bid, similarly for (1b') for its exit time. Constraint (1c') ensures that at most one bid is accepted per agent (exclusive-or bid constraints), and that no bids are accepted from dropped trains.

Price Updates

Each dispatcher agent maintains ask prices on a discrete price lattice, but without imposing a discretization on the times that a train agent can bid. Ask prices represent a lower-bound on the price that a train agent must bid to have any chance of success in the auction, but do not guarantee that a bid will be successful. The lattice structure is used to approximate a continuous non-linear price space. A smaller unit of discretization leads to a higher computational cost and slower convergence but perhaps to a higher schedule quality. An alternative price structure might explicitly maintain unsuccessful bids and compute ask prices on-the-fly exactly.

Dispatcher agents provide a price-query function for train agents, to allow train agents to compute the ask price for any pair of times on-the-fly. Given a bid for a pair of fixed times the ask price is determined as the price of the nearest point in the lattice. Given a bid containing one or more flexible times, the ask price is determined as the *minimal* price over all lattice points with *consistent* times. The interpretation of consistent is the natural one, a lattice point is consistent with a bid if it satisfies all constraints on entry and exit times.

The *minimal* operator, used to compute ask prices across sets of lattice-times consistent with flexible bids, provides useful price semantics:

$$p(k_1) \geq p(k_2), \quad \text{if } k_1 \subseteq k_2$$

for bids k_1 and k_2 if all times consistent with k_1 are also consistent with k_2, and the bids have both flexible entry and exit times. This follows immediately from the minimal operator. The minimal price over the set of times consistent with k_2 can be no larger than the minimal price over the set of times consistent with k_1, because the set of consistent times for k_1 is a subset of the times for k_2. The relationship allows a train agent to prune its local search when considering different times in its best-response strategy.

Note, however, that it is *not* necessarily the case that $p(k_1) \geq p(k_2)$ for a pair of *fixed* times k_1 and k_2 with the entry time of k_1 later than k_2 and the exit time of k_1 earlier than k_2; for example, bid k_2 might hit a *safety* conflict with an accepted bid from another agent, which bid k_1 might escape. In comparison, a flexible bid, k_2, would escape the safety problem whenever k_1 escaped the safety problem.

An unsuccessful bid increases the price on its nearest lattice point, or multiple consistent lattice points in the case of an unsuccessful constraint-based bid. For each bid in an unsuccessful exclusive-or set of bids we update the ask price on all grid points consistent with the bid times, as follows:

(a) if the bid is for fixed times then find the point on the lattice closest to the bid, otherwise find the set of consistent points,

(b) update the ask price at that lattice point to ϵ above the unsuccessful bid price, or on all lattice points in the set, where $\epsilon > 0$ is the minimal bid-increment in the auction.

The structure of this price-update is motivated by price updates in *i*Bundle (Chapters 4–7), which increases prices on bundles of items by ϵ in response to unsuccessful exclusive-or bids.

The price is also increased because of bids submitted by train agents in the provisional allocation that receive the same pair of times from the last round of the auction but are trying to shift away from that allocation. I allow a train agent to indicate when it is merely repeating a bid because that is required under the auction rules, rather than because it really wants that pair of times.

Finally, an "infinite" value is used to represent the case that the *safety* condition will be violated with any bid close to a particular grid point. This is used as items are sold to train agents at particular times (under the continuous clearing rules, see below), to move a train's bid focus away from a time that cannot be accepted at any price.

Bidding Rules

The bidding rules are quite simple: (1) an agent must bid at least the ask price for a pair of times, computed as appropriate for the flexible/fixed attributes of the bid; and (2) an agent must repeat a bid that supports a pair of times it receives in the current provisional allocation. The two rules ensure that progress is made across rounds of the auction towards a solution.

Clearing and Termination Rules

Every dispatcher auction terminates simultaneously when no new bids are placed by any train agent to any dispatcher agent. In addition, the auctions have a continuous clearing rule, in which a dispatcher commits to a particular pair of times for an agent that has received the times in the provisional allocation for more than a fixed number of successive rounds, T_{clear}.

Continuous clearing helps to reduce bidding complexity, committing trains to particular times (although they can continue to bid for alternate times at an additional cost), and focusing their search. A countervailing force is that early commits can also lock-in a particular pair of times too quickly when continued search might find a better solution.

The MIP formulation for winner-determination is easily adapted to include committed times. These times can be represented with bids from a dummy agent, with acceptance of those bids forced within the MIP solution method.[4]

Speeding-up Winner-Determination

One useful technique to speed-up winner-determination in iterative auctions is to maintain solutions from previous rounds in a cache, indexed against the bids that were submitted. The cache can be checked for a solution before solving the mixed integer program.

A *hit* in the cache depends on the bid times and fixed/flexible attributes in agents' bids. We need a couple of definitions.

Definition 9.1 [less flexible] A bid k_1 for a pair of times at a price is weakly *less flexible* than another bid k_2 if all times that are consistent with bid k_1 are also consistent with bid k_2, and if the price on bid k_1 is no greater than the price on bid k_2.

In other words, a bid k_1 which is less flexible than another bid k_2 will never be accepted in the provisional allocation when bid k_2 is not accepted.

Definition 9.2 [supports] A set of bids *support* a bid k' if one or more of the bids in the set is (weakly) more flexible than bid k'.

In otherwords, the agent that submitted the set of bids would have been happy to submit a successful bid k'.

[4] The flexibility of mixed-integer program formulations of winner-determination problems was previously noted by Andersson *et al.* [ATY00].

Given this, a set of bids received by a dispatcher *match* a set of cached bids if there is some permutation of the new bids with a set of bids in the cache that satisfies:

(1) if bids from agent i and are successful in the cached solution, then the new bids from i must *support* the time corresponding to the successful bid, and be (weakly) *less flexible* than the other times in the cached bid.

(2) if bids from agent i are unsuccessful in the cached solution, then the new bids from i must all be (weakly) *less flexible* than the old bids.

It is quite straightforward to understand why this cached solution is an optimal solution with the new bids. Basically, no agents that are in the provisional allocation in the cached solution submitted stronger bids on anything except the successful bid, and no unsuccessful agents submit any stronger bids.

As a special case, a match also occurs when the new bids from every agent supports a successful bid in a cached solution, and the prices on all the rejected bids from each agent are no greater than on the accepted bids. In this case the rejected bids from an agent do not need to be less flexible than the other times in the cached bid.

9.4 The Bidding Problem

Recall that each train $i \in \mathcal{I}$ has value V_i to complete its journey, subject to a cost $cost_i(t_\mathrm{s}, t_\mathrm{d})$ for off-schedule performance, given optimal source and destination times $t^*_\mathrm{s}(i)$ and $t^*_\mathrm{d}(i)$ and actual times t_s and t_d. The bidding problem is to purchase the right to travel across the network from source to destination at minimal total cost, where cost is the sum of the price it pays in each auction and the cost of off-schedule performance. In addition, if this cost is greater than the train's value then it would prefer to drop out completely.

The bidding problem is difficult for two main reasons:

(a) *coordination:* a train agent must bid with multiple dispatchers when its route spans more than one territory.

(b) *dynamic pricing:* a train agent cannot know the prices at which the auction will clear.

Each train agent is assumed to follow a myopic best-response bidding strategy, bidding for the schedule that minimizes total cost given the current ask prices. Myopic best-response provides a good starting point to analyze the performance of the auction method. It would be interesting, but probably quite difficult, to also consider the effect of fully

strategic agent behavior on the quality of scheduling solutions. It is possible that a train agent can achieve a better outcome by anticipating the bids of other agents and considering the effect of current bids on future prices.

9.4.1 Myopic Best-response Bidding Strategy

The myopic best-response bidding problem can be formulated as a *shortest weighted path problem*. The edges in the graph correspond to pairs of entry-exit times at each dispatcher, fixed or flexible as appropriate. Edges are connected if the exit time on one edge is *compatible* with the entry time on the next edge. The compatibility requirement here simply requires that there is enough time for the train to leave the first dispatcher territory at the exit time, travel across the connecting yard, and enter the second dispatcher territory at the entry time. The cost associated with an edge represents the sum of the current ask price, and any cost for off-schedule arrival or departure if the dispatcher is at the source or destination of a train's route.

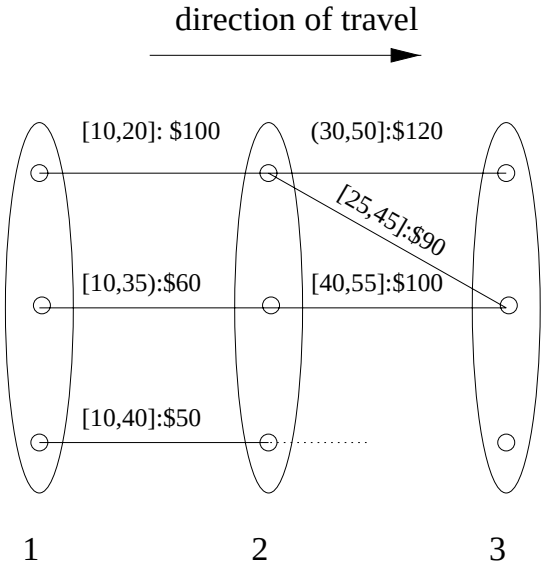


Figure 9.4: The myopic best-response bidding problem.

Figure 9.4 illustrates a partially-completed graph, with nodes *1, 2* and *3* representing the yard to the West of dispatcher d_1, between d_1 and d_2, and to the East of d_2 respectively. For example, the times on the edges between yards *1* and *2* denote entry-exit times for dispatcher d_1, together with the ask price for those times, summed with any additional cost of early/late departure if dispatcher d_1 is also the first dispatcher in the train's itinerary.

Formulation

A train's best-response, taking current prices as fixed, is to select a path from source to destination with minimal total cost (or no path at all in the case that the minimal cost is greater than its value for completing the journey).

Given a set of dispatchers, $\mathcal{D}$, let $(d_1, \ldots, d_n)$ represent the dispatchers on the route of a particular train. Let $C^*_{1\to n}(t)$ denote the minimal total cost to enter dispatcher d_1 no earlier than time t, travel from d_1 to d_n, and exit from dispatcher d_n. This cost represents the cost of the best schedule, given current ask prices and the train's costs for off-schedule performance. The solution to C^* can be computed as a recursive relationship:

$$C^*_{j\to n}(t) = \begin{cases} \min\limits_{\tau>t} \left(c_j(t,\tau) + C^*_{(j+1)\to n}(\tau) \right) & \text{, if } j < n \\ \min\limits_{\tau>t} c_j(t,\tau) & \text{, if } j = n \end{cases}$$

where $c_j(t_1, t_2)$ is the cost to enter dispatcher j at time t_1 (or no earlier than t_1 in the case of a flexible bid time), and exit dispatcher j at time t_2 (or no later than t_2 in the case of a flexible bid time), computed as the sum of the price for times and any additional cost penalty for off-schedule performance if dispatcher j is at the end of the train's route.

The price is the ask-price if the agent is not yet committed to the good (i.e. it has not cleared), or *zero* otherwise (in which case the price represents a sunk cost). Trains treat offered items in the same way as any other item, making an assumption that it can move away from such an item if necessary without becoming exposed.

Trains consider flexible bid times in the case of non-extremal nodes, but fixed times at source or destination because a cost is incurred for any deviation from optimal departure and arrival times. The intermediate time τ represents the time to cross from dispatcher d_j to d_{j+1}. In this description I have finessed the detail about the time to travel across yards between dispatch territories, which is simply incorporated into the recursion.

Dynamic Programming

Dynamic programming solves this shortest-path formulation of the myopic best-response bidding problem, computing the best solution over a fixed set of time points selected by the train agent, and working from dispatcher d_n to d_1, pruning any dominated solutions (for example higher cost edges with earlier entry times). In related work, Boutilier *et*

al. [BGS99] proposed a dynamic programming algorithm for agent bidding strategies in *sequential* auctions with complementarities.

We implement the following algorithm:
(1) determine the maximal *compatible* set of current offers and sold items, that leave enough time for travel across the dispatch territories.
(2) for each maximal set, use dynamic programming to determine minimal-cost routes in the gaps of the schedule. The gaps are contiguous sequences of dispatchers for which the train is not currently holding a suitable pair of entry and exit times. Flexible time constraints are selected for all entry and exit times except those representing a train's initial departure or arrival time, at which nodes the train is not willing to be flexible.
(3) fill the gaps and select the solution with the lowest total cost (including the cost for current offers/sold items used in the solution).

Compatible offers allow the train to complete its journey, including the time to travel across dispatchers and across connecting yards, consistent with all entry and exit times in the offers. A maximal set of compatible offers is a set of compatible offers with maximal cardinality, given the current offers and the train's free-running travel times.

Whenever a gap occurs at the start or end dispatcher on a train's route the train agent also considers tradeoffs between bid price and the cost of off-schedule performance for off-time departure and/or arrival times.

This method includes a bias in favor of solutions compatible with times the train receives in the current provisional allocations. This is reasonable, given that the ask prices represent a lower-bound on what might be a successful bid price but provide no guarantees that a bid will actually succeed. That an agent currently receives a pair of times conveys useful information about the fit of those times with bids from other agents.

Finally, a train will submit as many bids that are consistent with its selected solution as possible, making use of multiple exclusive-or bids with individual dispatchers, and using constraints on times to submit multiple bids without compromising the solution. This increases its own likelihood of success, and also helps the dispatchers to coordinate joint search across multiple agents.

9.5 Experimental Results

The performance of the auction-based solution is compared with a centralized solution in networks consisting of *linear chains* of dispatcher territories. Both the global MIP formulation and the MIP for winner-determination in each round of the auction are solved with CPLEX, commercially available linear-programming based branch-and-bound optimization software. The rest of the code (myopic best-response, price-updates, etc.) was written in C++.

9.5.1 Dispatcher model

Each dispatcher territory has the same simple network structure, as depicted in Figure 9.5. The total distance is 157.5 km, consisting of a single-track section followed by a double-track section (siding) followed by a single-track section. Dispatcher territories are connected together with yards. Trains have a maximal speed of 100 km/hr over single- and double-track sections, and 1 km/hr within a yard. For simplicity, I model maximal speed as 100 km/hr throughout the network and re-scale yards from size 0.5 km to a model length of 50 km. The minimum separation distance, Δ_{safety}, between trains is 20 km, corresponding to a run-time of 0.2 hrs at maximum speed.

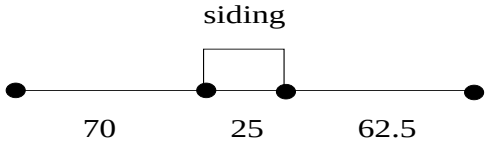
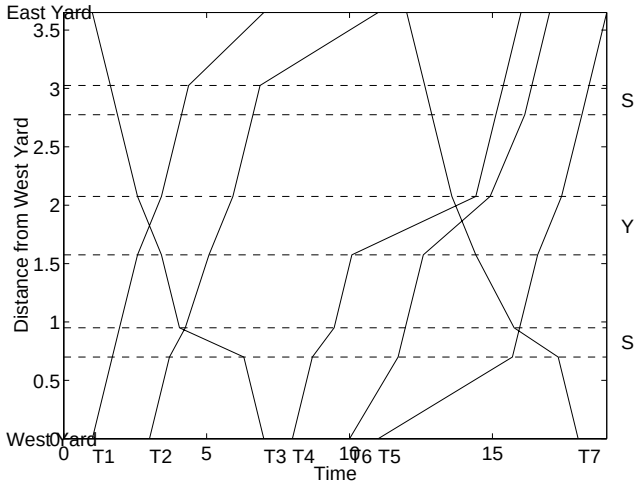


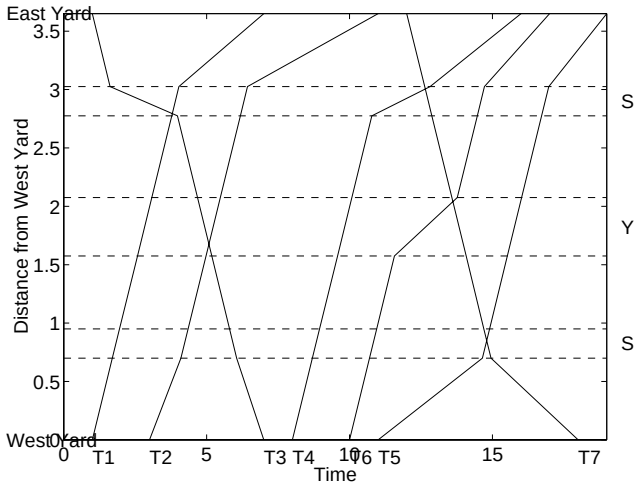
Figure 9.5: The network structure for a single dispatcher, with distances of each section (in km).

9.5.2 Example Problem

Consider a problem with a chain of two dispatchers, and 7 trains, each with value \$200 and cost \$50 per hour of delay. Trains 1, 2, 4, 5, and 6 run east with optimal departure and arrival times (in hours) of { (1, 7), (3, 11), (8, 16), (11, 19), (10, 17) }, while trains 3 and 7 run west with optimal times of { (1, 7), (12, 18) }. Given a maximal speed of 100 km/hr the *free-running time* of a train across the network is 3.66 hrs. This is how long a train would take with no delays.



(a) Auction solution.



(b) Centralized solution.

Figure 9.6: Example: 7 trains, 1 dispatcher territory. Distance in 100's of kms, time in hrs.

The auction-based and centralized solutions to this problem are illustrated in distance-time charts in Figure 9.6. Both methods find optimal solutions, with value \$1400 (all trains run on-time). This problem is quite under-constrained, with a number of possible optimal schedules.

Notice that the auction-based solution is less extremal than the centralized solution. This is quite typical, a result of the fact that train agents tend to bid less extreme times than those selected with a global LP-based branch-and-bound method such as CPLEX, and achieve a more evenly paced schedule from source to destination. We would expect this property to make auction-based solutions more robust against unexpected minor delays during the execution of a schedule.

9.5.3 Results

The experimental results are for problems with between 2 and 4 dispatcher agents, and between 5 and 15 train agents, and with linear networks— formed from dispatcher territories as shown in in Figure 9.5 and connected with yards. The auction algorithm was parameterized as follows: at most 5 bids per-agent in each round, at most 240 seconds to solve winner-determination in each round (with the best feasible solution used if the optimal solution is not found), and a minimal price increment of \$25. The price lattice was maintained over points with a granularity of 0.2 hrs, and I used a time interval size of 0.3 hrs in the trains' dynamic-programming algorithm.

Problem Generator

A stochastic model is used to generate a set of problem instances. The model is loosely based around instances in Kraay et al. [KHC91]. I refer the reader to Hallowell [Hal93] for an account of other interesting train scheduling problem sets in the literature. The problem sets are parameterized by constants: prob(E), μ_V, σ_V, μ_C, σ_C, $dep_{\max}$, μ_{slack} and σ_{slack}, and the number of train and dispatcher agents, as described below

All trains travel from end-to-end over the network, and travel East with probability prob(E). A train's value is selected from a normal distribution, $V_i \sim N(\mu_V, \sigma_V)$, with mean μ_V and standard deviation σ_V, and its unit cost C_i for off-schedule performance is normally distributed $N(\mu_C, \sigma_C)$. The optimal departure time for a train, $t^*_{\mathrm{d}}(i)$, is uniformly distributed, $t^*_{\mathrm{d}}(i) \sim U(0, dep_{\max})$. Finally, a train's optimal arrival time, $t^*_{\mathrm{s}}(i)$, is computed

	Model Size								
num dispatchers	2			3			4		
num trains	5	10	15	5	10	15	5	10	15
Global-time (s)	97	1438	2022	1161	2442	2495	886	2378	3155
Global-value ($)	895	1699	2097	854	1561	1177	898	1106	1132
Auc-time (s)	15	792	2568	15	1192	2039	26.9	944	2448
Auc-value ($)	850	1893	1737	842	1855	2632	768	1832	2162
Agent time (s)	0.4	0.6	2.7	0.9	2.2	3.3	1.9	4.4	8.8
Revenue ($)	315	698	1045	470	1030	1690	700	1365	2142
num rounds	12	13	25	13	16	18	16	16	23
Cache hit (%)	60	50	30	61	50	47	50	52	37

Table 9.1: Comparative performance: Auction vs. Centralized methods.

so that the *relative slack*, i.e. (available time - free-running time) / free-running time, is normally distributed with mean μ_{slack} (%) and standard deviation σ_{slack} (%).

The complexity of a train-scheduling problem depends on many factors, including the slack time available to each train, the network section types, and the number of "cross-overs". A *cross-over* is counted whenever two trains traveling on-time must cross at some point in the network. As I scaled the problems, with more train agents and more dispatcher agents, I adjusted the $dep_{\max}$ parameter to maintain the same number of average cross-overs per-agent, in an effort to maintain a similar problem complexity. An appropriate $dep_{\max}$ value was computed off-line for each problem size to achieve this property. Without this adjustment adding more trains and more dispatchers increases the number of cross-overs and makes problems much more difficult to solve. My goal was to capture the scaling properties of the two solution methods with the same per-train and per-dispatcher complexity.

Results

In the experiments the stochastic problem generator is parameterized with: prob$(E) = 0.7$, $\mu_V = \$200$, $\sigma_V = 50$, $\mu_C = \$100$, $\sigma_C = 25$, $\mu_{slack} = 100\%$, $\sigma_{slack} = 25\%$, and set $dep_{\max}$ to give average cross-over complexity of 2 per-train. I generated 10 problem instances for each problem size, as the number of train agents were increased from 5 to 15 and the number of dispatchers from 2 to 4.

Table 9.1 presents the results. The computation time of the centralized method is bounded at 3600 secs, at which point we take the best available solution. CPLEX also

ran out of memory (at 200 MB) while solving a few of the centralized problems, stopping before 3600 secs but without an optimal solution. The computation time of the auction is the total winner-determination and price-update time, summed over all rounds and all dispatchers.

Notice that the quality of the schedule computed in the auction dominates that from the centralized solution in hard problems, as the number of dispatchers and/or the number of agents increase— the auction computes a higher quality solution in less time. The auction also appears to have reasonable scaling properties, with the number of train agents and in particular with the number of dispatchers. We believe that this advantage arises because the auction is an effective method to decompose the computational problem across dispatchers, with critical (and difficult) computation localized to a single critical dispatcher. It is also noteworthy that the train agents' myopic best-response algorithm is quite fast, indicating that it would be interesting to experiment with a smaller discrete time step. Notice also that the simple cache proves quite effective, finding the optimal solution around 50% of the time.

More experiments are required, both to better understand the average-case scaling properties of the auction and also to look more deeply at agent strategies. Based on these limited results my conjecture is that the average-case run time in the auction scales *quadratically* with the number of train agents, and perhaps *sub-linearly* with the number of dispatch territories. It would be interesting to develop an analytic model to confirm/reject this conjecture.

In terms of agent strategies, I suspect that some agents purchase times that they cannot use as the number of dispatchers increases and as the bid coordination problem gets more difficult. This belief is based on a comparison of the revenue with value in the auction, see Table 9.1. If this is the case it will be necessary to consider more sophisticated bidding strategies to avoid this exposure problem. A similar exposure problem is noted in the FCC spectrum auction problem, in which agents need sets of compatible licenses, and bid across simultaneous auctions [BCL00].

9.6 Related Work

This is not the first study of auction-based methods for train scheduling problems. Brewer & Plott [BP96], proposed the BICAP ascending-price auction for distributed train scheduling. The auction proposed in this chapter is more flexible: we allow trains to construct arbitrary schedules across the network, while BICAP restricts trains to bid from a small set of fixed paths. Returning to centralized approaches to train scheduling, Kreuger *et al.* [KCO97] have proposed a constraint-based method which appear to have better scaling properties than straight applications of MIP methods, but is perhaps less suited to making tradeoffs across schedules with different qualities.

Market-based methods have also been advocated for other distributed scheduling problems, such as for airport take-off and landing slot allocation problems [RSB82]. For example, Wellman *et al.* [WWWMM01] propose an auction-based method for a factory-scheduling problem, in which agents compete for periods of time on (one or more) shared machines. As in the train-scheduling problem, agents in the factory-scheduling problem (representing a job) often require a combination of time periods, perhaps also across multiple machines. The train scheduling problem differs in that it is not possible to define a *static* set of mutually-compatible times, any of which can be safely allocated to any agent. Instead, a feasible allocation of times must be checked dynamically, by computing a safe underlying meet/pass schedule. My approach is also rather different to that adopted by Wellman *et al.*, since I avoid imposing a discretization of time into finite slots, but provide instead a simple and expressive constraint-based bidding language.

9.7 Discussion

The auction-based mechanism for the distributed train scheduling problem presented in this chapter is a better match to the natural information and control structure of modern railroads than traditional centralized scheduling solutions. Moreover, initial experiments on a simple network structure show that the auction-based solution can generate better solutions than a centralized approach, and more quickly. The auction-based approach also appears to have useful scaling properties.

The main weakness in my current results is the lack of an interesting network structure. I assume a single-line network, which while quite a common assumption in the academic

train scheduling literature [KHC91, KH95, Hal93] is perhaps not very realistic. In the context of an auction-based method, relaxing the single line assumption would require two important extensions:

(a) extend the mixed integer program formulation of the winner-determination problem for each train agent to schedule trains across multiple lines,

(b) extend the bidding strategy of train agents to consider alternative routes, *in addition* to alternative times.

Extension (b) does not seem to be too difficult, given that train agents already consider alternative times (but not alternative paths); the current shortest path approach should extend quite readily. In addition, although it is not immediately clear how to formulate a multi-line problem as a mixed integer program, extension (a), the problem is certainly no more difficult than that faced in a centralized solution.

In addition to introducing heuristic methods and approximations for winner-determination in each round of the auction, it would also be interesting to compare the auction-based method with heuristic centralized methods. The current comparison with an integer-programming based centralized solver may be a little unfair, in that one can view the auction-based method (with prices increases, and myopic best-response, etc.) as combining special-case heuristic methods with a general-purpose integer-programming method.

Chapter 10

Conclusions

Auctions offer great promise as mechanisms for optimal resource allocation in complex distributed systems with self-interested agents. However, limited and costly computation necessitates a rethinking of traditional auction theory because direct extensions of auctions that work well in small problems can fail in complex distributed systems. My thesis is that it is necessary to take an explicitly computational approach to auction design. Indeed, the value of auctions in e-commerce systems will depend on the ability to maintain the desirable properties of auctions, for example economic efficiency, robustness, and simplicity, as methods are introduced to allow tractable computation. Once computational issues are successfully addressed, auctions may provide simple, stable, and robust solutions to many important distributed optimization problems.

Agents often demand bundles of items, and have values for bundles that are not equal to the sum of the values of the items in the bundle; e.g., for task allocation, procurement problems, and dynamic bandwidth allocation. Combinatorial auctions allow agents to represent these preferences explicitly by submitting bids on bundles of items, for example stating "I only want A if I also get B". Desirable properties of combinatorial auctions include strategy-proofness, such that truthful bidding is optimal whatever the strategies of other agents, and allocative-efficiency, such that the auction implements the value maximizing allocation. However there is a tension between the classic game-theoretic solution to this problem and computational efficiency.

The Groves family of mechanisms, and the Generalized Vickrey Auction (GVA) in particular, are strategy-proof and efficient. But the GVA is a sealed-bid combinatorial auction. Every agent must first report its value for every possible bundle, before the auctioneer

computes the allocation and agent payments. Sealed-bid auctions are undesirable computationally because of this complete revelation requirement, which soon becomes intractable in large complex domains for agents with hard valuation problems. Iterative mechanisms are absolutely critical in such domains because they can solve problems without complete information revelation from agents, and even without individual agents computing their exact values for all outcomes. The revelation principle of mechanism design, coupled with the uniqueness of Groves mechanisms provides useful guidance. Any strategy-proof and efficient iterative combinatorial auction must compute Groves payments at the end of the auction.

My dissertation develops an efficient iterative combinatorial auction, which avoids the centralization and complete revelation problems of the GVA. The auction, *i*Bundle, and its extension *Extend&Adjust*, is an ascending-price combinatorial auction that allows agents to adjust their bids in response to bids from other agents. *i*Bundle solves realistic problems without complete information revelation from agents, and with Extend&Adjust inherits much of the strategy-proofness of Groves mechanisms by computing Vickrey-Groves payments at the end of the auction. Instead of terminating as soon as the efficient allocation has been determined, Extend&Adjust remains open just long enough to collect additional information from agents (via best-response bids) to compute Vickrey payments.

In summary, the main contributions to computational mechanism design are:

- *i*Bundle Extend&Adjust, an iterative combinatorial auction that computes minimal competitive equilibrium prices in the combinatorial allocation problem, with myopic best-response agent strategies.

- *i*Bundle Extend&Adjust, an iterative combinatorial auction that computes the efficient allocation and Vickrey payments in all problems in which Vickrey payments can be priced in (non-linear and perhaps non-anonymous) competitive equilibrium, with myopic best-response agent strategies.

Another significant contribution is:

- VickAuction, a primal-dual algorithm that computes Vickrey payments with myopic best-response information from agents.

*i*Bundle Extend&Adjust interprets the primal-dual algorithm VickAuction as an

ascending-price auction. The difference introduced in *i*Bundle Extend&Adjust is an explicit method to adjust prices in the second "extend" phase of the auction, which aims to be more "natural" to participants than the method in VickAuction. An agent's bids in the rounds that follow the normal termination point of *i*Bundle affect neither the final allocation nor the final price that the agent pays. The only effect is to reduce the price that other agents pay. Therefore it is important that an agent should not be able to detect that the auction is in this phase, or else it would simply drop out and wait for the auction to finally terminate.

The outstanding challenge is to complete a proof of the following conjecture.

- (conjecture) *i*Bundle Extend&Adjust is the first iterative combinatorial auction to compute the efficient allocation and Vickrey payments in all combinatorial allocation problems.

We provably compute Vickrey payments and the efficient allocation whenever the extended auction terminates, but still need a proof that the current dummy agent rules are sufficient for termination. There is very strong experimental evidence that this is indeed the case.

The combined system, *i*Bundle Extend&Adjust, with a proxy bidding agent interface, is a promising dynamic mechanism to solve combinatorial allocation problems with distributed self-interested agents. The proxy bidding agents constrain agents to follow myopic best-response strategies, boosting the robustness-to-manipulation that is achieved through terminating with Vickrey payments.

10.1 A Brief Review

Chapter 1 provides an introduction to the computational problems inherent in classic game-theoretic mechanisms, such as the Groves mechanisms. The Groves mechanisms are very centralized solutions to decentralized optimization problems. The mechanism makes truth-revelation a dominant strategy for agents, but requires them to provide complete information about their preferences, to the auctioneer which then computes an optimal system-wide solution.

An important challenge in computational mechanism design is to develop a strategy-proof and dynamic mechanism, in which it is an agent's dominant strategy to provide

truthful information incrementally to the mechanism, until just enough is known about agents' problems to solve the system-wide problem. The scheme proposed in this dissertation, comprising of *i*Bundle Extend&Adjust, makes significant progress in this direction for the combinatorial allocation problem. Transformation techniques between the exclusive-or bidding language provided in *i*Bundle and other more expressive bidding languages also promise to lead to domain-specific implementations that can exploit structure in agent preferences without incurring the potentially exponential cost of a translation into explicit values on bundles.

Chapter 2 introduces important results from classic mechanism design, including the Groves family of mechanisms. Groves mechanisms are the *only* strategy-proof and efficient mechanisms among direct-revelation mechanisms. Taken along with the *revelation principle*, which states that any mechanism can be implemented as an equivalent direct-revelation mechanism (with agents reporting complete information about their preferences), this uniqueness provides useful focus to the design of iterative mechanisms. In particular, any efficient and strategy-proof iterative mechanism must compute the outcome of a Groves mechanism for the underlying preferences of agents.

The revelation principle does not mean that we need to only consider direct-revelation mechanisms when constructing mechanisms for combinatorial problems. The revelation principle assumes unlimited computational resources, both for agents in submitting valuation functions, and for the auctioneer, in converting a mechanism into a single-shot direct-revelation solution. In fact, the work in my dissertation clearly demonstrates that iterative auctions *expand* the implementation space when computation is an issue.

The tensions between classic game-theoretic solutions and tractable computational solutions soon become evident as one considers the application of mechanisms to difficult distributed optimization problems, such as supply-chain procurement or bandwidth allocation. Chapter 3 considered the computational demands within a mechanism, addressing computation at both the infrastructure (e.g. auctioneer) and agent level.

The first important cost is the *computational complexity* on the mechanism infrastructure. I surveyed a number of methods to address this cost, including approximations, special-cases and decentralized methods. Naive attempts to introduce approximations into mechanisms can break useful game-theoretic properties, such as strategy-proofness.

The second important cost is the *strategic complexity* on agents, which is closely linked

to its game-theoretic properties. In particular, a mechanism in which every agent has a *dominant strategy*, i.e. an optimal strategy whatever the strategies and preferences of other agents, is useful game-theoretically *and* computationally. From a game-theoretic perspective, a dominant strategy equilibrium is a very robust solution concept, stable for example against "irrational" agents. From a computational perspective, with a dominant strategy an agent can avoid costly modeling and game-theoretic reasoning about other agents.

The third important cost is the *valuation complexity* on agents; valuation problems are often hard, agents have limited and/or costly computation, and there are many different outcomes in combinatorial domains. I considered two approaches: high-level bidding languages and dynamic methods.

High-level bidding languages, such as *bidding programs*, can help to address valuation complexity in situations in which it is easier to specify a method to compute values for outcomes than to compute values for every possible outcome. This might be the case if an agent has a well-formulated optimization problem to compute its value for different resource bundles. However, it is quite likely that each outcome has a different structure, and requires additional information and a new specification. Another problem with this approach is that it shifts computation to the center and is problematic from a privacy and informational perspective.

My dissertation suggests *dynamic mechanisms* as an alternative approach to address valuation complexity, in which agents reveal incremental information about their preferences until the mechanism can compute and verify an optimal solution. *i*Bundle is an iterative combinatorial auction that solves realistic problems without complete information revelation from agents. The idea is to "open up" the algorithm for computing the outcome of the GVA, and involve agents dynamically in the computational process. It is easy to construct examples in which it is not necessary to have complete information about agents' valuation problems to compute and verify the outcome of the auction. Some simple examples were described towards the end of Chapter 3. A well structured dynamic method will ask agents for just enough information to enable the mechanism to compute and verify the outcome.

Table 4.7 summarized the progress in iterative auction design over the past two decades.

Each contribution relaxes assumptions on agent preferences and/or strengthens the equilibrium analysis of the auction. All auctions terminate with efficient allocations and prices that are in competitive equilibrium, or equal to Vickrey payments, or both. Prior to *i*Bundle there was no method that terminated in competitive equilibrium in the general problem, let alone in the minimal competitive equilibrium solution, or with Vickrey payments.

My approach to iterative combinatorial auction design integrates principles from linear programming and game-theoretic mechanism design.

First, I assumed that agents will follow a myopic best-response bidding strategy in every round of the auction. This allowed the direct application of linear programming theory, and primal-dual algorithms in particular, to auction design. With myopic best-response there is an easy mapping from an suitable primal-dual algorithm to an ascending-price combinatorial auction. The corresponding auction, *i*Bundle, is the first efficient and iterative combinatorial auction for any reasonable agent bidding strategy, in this case myopic best-response.

Second, I extended the approach to compute Vickrey-Groves payments at the end of the auction, in addition to the efficient allocation. When successful, this makes myopic best-response an optimal *sequential* strategy for an agent in the auction, in equilibrium with best-response from every other agent. I derived a new primal-dual algorithm, VickAuction, from a linear program formulation of the Groves mechanism for the combinatorial allocation problem. VickAuction provably computes the Vickrey payments with only best-response information from agents. VickAuction corresponds to *i*Bundle with a second phase, Extend&Adjust. The purpose of the second phase is to elicit enough additional preference information from agents to compute Vickrey payments. It turns out that Vickrey payments require *more information* than that which is required to compute the efficient allocation. Vickrey-Groves payments are computed by an adjustment procedure from final prices, based on primal-dual information collected from best-response bids from agents.

Chapters 4 and 5 presented the *i*Bundle auction, which computes efficient allocations with agents that follow myopic best-response bidding strategies. Chapter 4 reviews linear program models for the combinatorial allocation problem, and introduces a primal-dual algorithm, CombAuction, that dynamically computes prices with enough structure to support the efficient allocation in competitive equilibrium. In some problems it is necessary

to use non-anonymous prices, with a different price for the same bundle to different agents, and non-linear prices, with the price on a bundle different from the total price over the items in a bundle.

CombAuction has a natural interpretation as an ascending-price auction. The primal solution corresponds to a provisional allocation and the dual solution corresponds to bundle prices. Complementary slackness conditions demonstrate that a provisional allocation is efficient whenever: (1) every agent receives a bundle in its myopic best-response bid set, and (2) the allocation maximizes revenue for the auctioneer. This connection to linear programming theory proves the allocative-efficiency of *i*Bundle.

Experimental results confirm that *i*Bundle computes efficient allocations across a suite of problem instances with myopic best-response agent strategies, and with less information revelation than in the GVA. In addition, the effect of price discrimination on allocative efficiency is small, and I expect *i*Bundle to perform well with anonymous prices in most problems. Approximations are possible within *i*Bundle. For example, increasing the minimal bid increment ϵ in *i*Bundle decreases the number of rounds and can provide an order-of-magnitude speed-up for small losses in allocative efficiency. Methods to leverage the sequential nature of winner-determination within *i*Bundle were also studied, via caching and hot-start techniques.

Chapters 6 and 7 introduced *Extend&Adjust*, a method to extend *i*Bundle and compute Vickrey payments at the end of the auction, in addition to the efficient allocation. The method justifies myopic best-response, making myopic best-response an optimal sequential strategy for an agent in equilibrium with best-response strategies from other agents. The design leaves *i*Bundle basically unchanged, simply keeping the auction open for a few more rounds and then computing discounts to agents at the end of the second phase. In the end, agents are charged discounted prices based on discounts computed during the second phase.

First, I derived a linear program to compute *minimal* competitive equilibrium (CE) prices from a set of suitable competitive equilibrium prices and the efficient allocation. Minimal CE prices provide some protection against manipulation, and support the Vickrey payments in special cases. The linear program formulation leads to the procedure Adjust, which computes minimal CE prices from the price information at the end of

CombAuction, and similarly at the end of *i*Bundle. I characterized necessary and sufficient conditions under which Adjust will compute minimal CE prices, and proved that *i*Bundle(3) (the variation with non-anonymous prices in all rounds) with Adjust terminates with minimal CE prices. The discounted prices are equal to Vickrey payments in all problems for which minimal CE prices support Vickrey payments.

Theorem 10.1 i*Bundle and* Adjust *is an iterative Vickrey auction when minimal CE prices support Vickrey payments.*

A fast but accurate method AdjPivot computes approximate discounts to agents after *i*Bundle terminates, checking complementary-slackness conditions only with respect to the "pivotal" allocations that were previously computed as provisional allocations during the auction.

Second, I derived a linear program formulation to compute the Vickrey payment of any one agent in all combinatorial allocation problem instances. The Vickrey payment to every agent can then be solved as a *sequence* of independent linear programs. This leads to a linear program to compute the Vickrey payment to an agent from a set of suitable competitive equilibrium prices and knowledge of the optimal allocation (but without additional information about agent valuation functions or the value of the optimal allocation). Finally, I proposed procedure Adjust*, a slight variation on Adjust, as a method to compute Vickrey payments from the price information at the end of an iterative auction. Considering necessary and sufficient conditions on CE prices for Adjust* to compute minimal Vickrey payments, the most important non-obvious condition can be stated as follows:

... if an agent in the optimal allocation is not in one or more second-best allocations (without one of the agents in the allocation) then the price on the bundle it receives in the optimal allocation must equal its value.

This condition will not necessarily hold at the end of CombAuction, or at the end of *i*Bundle. However, I was able to derive a primal-dual algorithm, VickAuction, which computes Vickrey payments with Adjust*, by proposing a second phase to CombAuction, called PhaseII, which continues to adjust prices until the conditions for Vickrey payments are met. VickAuction implements the allocation computed at the end of

CombAuction, with adjusted prices based on discounts computed during PhaseII.

I prove optimality for VickAuction:

Theorem 10.2 VickAuction *is a primal-dual algorithm to compute the Vickrey payments and efficient allocation in the combinatorial allocation problem, with only best-response information from agents.*

Significantly, VickAuction, computes the Vickrey outcome with only best-response information from agents and without direct information about agent valuation functions.

Chapter 7 introduces an experimental auction method, *i*Bundle Extend&Adjust, to implement VickAuction in a decentralized system. The auction introduces a second phase to compute Vickrey payments, collecting just enough additional information from best-response agent strategies. The main difficulty in implementing the primal-dual method, VickAuction, as an auction arises because it is important that agents can not detect the transition from Phase I to Phase II. The experimental method used to drive price competition in the extended phase of *i*Bundle introduces *dummy agents* into the auction as real agents drop out, to: (a) provide enough competition to drive the prices to agents in the efficient allocation high enough to compute Vickrey discounts to every agent; and (b) provide a "competitive effect" that is hard to distinguish from the bids of the real agents they replace. The valuation functions of the dummy agents are configured by the auctioneer dynamically to mimic continued bidding from the real agents as they drop out.

I proved that the adjusted prices in *i*Bundle Extend&Adjust are Vickrey payments when the auction terminates. The outstanding open question is whether the current rules for quiescence detection and dummy agents are sufficient to generate termination conditions in all problems.

Chapter 7 presented encouraging experimental results for *i*Bundle Extend&Adjust. The extended auction is indeed able to compute the Vickrey outcome with myopic best-response agent strategies across a suite of problems. I make the following conjecture:

Conjecture 10.1 i*Bundle Extend&Adjust is an iterative Generalized Vickrey Auction.*

Vickrey payments make myopic best-response becomes a Bayesian-Nash equilibrium of the auction.

THEOREM 10.3 *Myopic best-response is a Bayesian-Nash equilibrium of an iterative auction that myopically implements the outcome of the Generalized Vickrey Auction.*

In other words, myopic best-response is a sequentially rational strategy for an agent in such an auction, in equilibrium with myopic best-response strategies from every other agent.

Finally, proxy bidding agents are suggested as a method to boost the strategy-proofness provided by Vickrey payments. The proxy agents act as an interface between the auction and the agents, receiving incremental information from agents about their preferences over allocations, and following a myopic best-response strategy with that information. The proxy agents check that the information provided in each round is mutually-consistent, and only bid when there is enough information to determine the best-response.

Given proxy agents we have the following proposition:

PROPOSITION 10.1 *Dynamic truth-revelation is a dominant strategy to the proxy agents if: (a) the proxy agents can constrain agents to providing information consistent with a single* ex ante *fixed (but perhaps untruthful) valuation function; and (b) the auction implements an iterative GVA with myopic best-response.*

The proxy agents cannot actually limit agents to a single valuation function, but checking consistency across rounds should be quite effective at constraining any possible manipulation. The only gap that remains in the auction's strategy-proofness is the extent to which consistency cannot be completely enforced without falling back on complete information.

Chapter 8 proposed a new auction property, *bounded-rational compatibility*, to characterize auctions in which agents can compute equilibrium strategies with approximate information about their preferences. For example, an iterative auction is *myopic* bounded-rational compatible if an agent can compute its myopic best-response strategy with an approximate valuation function, for example bounds on its value, in some problems. *i*Bundle is (myopic) bounded-rational compatible, while the GVA is not (dominant-strategy) bounded-rational compatible. The extended *i*Bundle auction is certainly Bayesian-Nash bounded-rational compatible in problems in which it computes Vickrey payments with myopic best-response strategies, because myopic best-response is the equilibrium strategy

in this case.

To begin to understand the advantage of myopic best-response over complete revelation I compared the efficiency and computation in iterative and sealed-bid auctions for a simple model of a bounded-rational agent. I modeled the deliberation decision of an agent explicitly, and computed myopically-rational metadeliberation strategies for agents in iterative and sealed-bid auctions. The experimental results showed that: (a) iterative auctions compute more efficient solutions than sealed-bid auctions, with agents allocating limited computational resources to more "important" bundles; (b) iterative auctions allow agents to avoid value computation.

Finally, Chapter 9 presented an application of auction-based methods to a distributed train scheduling problem. Auction methods are well suited to the natural information and control structure of modern railroads. In the model, trains bid for times to enter and exit the territories of each dispatcher along their route, while each dispatcher operates an ascending-price auction, for the right to enter and exit its territory at a particular time. *i*Bundle style price-update rules are applied to adjust the price on pairs of entry and exit times across rounds. One innovation is that train agents can bid with an expressive constraint-based bidding language to represent indifference across times, e.g. "any arrival time before 12 pm is fine", and without an imposed discretization on times. Computational results on a simple linear network, for train agents with myopic best-response bidding strategies demonstrated that the auction-based method computed better solutions than a centralized method and in less time and suggest that the auction method has useful scaling properties.

10.2 Future Work

Let me outline some areas for future work.

10.2.1 Iterative Combinatorial Auction Extensions

Expressive Bidding Languages

*i*Bundle provides agents with a complete, but not very expressive, exclusive-or (XOR) bidding language, which allows agents to bid for multiple bundles of items and specify they want at most one bundle. Expressive languages, for example constraint-based and

functional-approximation languages, can reduce the informational and computational demands on agents and lead to faster winner-determination algorithms.

One interesting approach to develop a mechanism for a new representation language is to understand how to adjust prices via a transformation of the representation into the *i*Bundle XOR/bundles representation. It may be possible to derive tractable rules for an allocatively-efficient dynamic mechanism in the new representation. First convert bids in the new language into the *i*Bundle XOR/bundle representation, then determine price-updates and the provisional allocation in the XOR/bundle representation, and then understand the rules for price-updates and rules for winner-determination in the new representation. Ideally we would like to exploit structure in agents' bids within the winner-determination and price-update problems in the auction, and *compile* this transformation procedure so that it is not necessary to work in the XOR/bundle representation when that is not useful computationally.

Consider a transformation from exclusive-or (XOR) to additive-or (OR) bids, as a simple example of this technique. Additive-or bids can be simulated in *i*Bundle as bids from separate agents. This illustrates that the *i*Bundle auction rules with additive-or agent bids (and without constructing explicit bid prices on bundles from agents' bids) are:

— increase prices on any single bundle in an OR bid for which an agent is unsuccessful in the current round

— never introduce price-discrimination

— terminate when every agent receives *all* bundles in its bid in the provisional allocation

Automated compilation methods could also be introduced, to allow a user to define a language semantics from which optimal auction methods are generated on-the-fly.

Application Study

One outstanding and important piece of experimental work is to perform a computational comparison of agent valuation complexity in *i*Bundle and the GVA in a combinatorial allocation problem domain, with concrete models for local agent problems. Here are some problems with desirable properties (hard agent valuation problems, well-formed central optimization problems, natural decentralization, existing problem sets, etc.).

- *Distributed traveling salesperson problem.* Andersson & Sandholm [AS98, AS99, AS00] define a multiagent TSP in which agents have locations and jobs have locations. The goal is to allocate jobs to agents to maximize the social welfare, which is measured as the total distance traveled by all agents. The authors report results for a distributed task allocation method on 8x8 simulations, with random agent and job locations selected on a simple grid.

- *Multiple project resource management.* Projects may involve dozens of firms and hundreds of people who need to be managed and coordinated. Examples include large construction projects, opening a new store, performing major maintenance, starting up a new manufacturing facility [SBG94]. One well-formulated multi-agent problem is known as multiple-project resource-constrained optimization, in which individual projects compete for the same constrained resources and must solve local scheduling problems to minimize costs given allocations. The local problem is hard, and as Shutb *et al.* [SBG94] note, it is not realistic to solve to optimality with several hundred activities.

- *Multi-agent scheduling problems.* A classic example is the airport takeoff and landing time-slot problem [RSB82, GIP89]. Airlines complete for takeoff and landing slots at airports. For a particular allocation of slots the airline must solve its local crew and airplane scheduling problem to compute the cost for using the slots.

Sequential Winner Determination

The winner-determination problem in iterative auctions presents an interesting dynamic computational problem, because agents' bids change only gradually during the auction as prices increase. In addition to simple caching across rounds, and hot-start techniques, one might also look to compute and re-use solutions to subproblems. As an example, with branch-and-cut optimization, in which new constraints are introduced during linear-program based branch-and-bound search, one can re-use cuts to prune search in later rounds. Another interesting approach is that of "continual computation", in which probabilistic and game-theoretic methods are used to predict future winner-determination problems and to precompute solutions in the down time between rounds.

The Agent Metadeliberation Problem

Given that an agent has a hard valuation problem, limited computational resources, and many possible bundles of items to value, how should an agent schedule deliberation across different bundles? This is an interesting domain for metadeliberation techniques, such as those that have found application in other time critical domains, such as in medical [Hor87] and robot path planning domains [BD89]. An iterative auction provides a dynamic and time-critical environment, with prices increasing across rounds, and a finite time delay between rounds. Values for bundles are nested within an agent's algorithm to compute its best response, and the expected utility of submitting a correct best-response in the current round depends on beliefs about strategies of other agents and future prices in the auction.

An Asynchronous *i*Bundle Auction

It would be useful to prove properties about *i*Bundle when some agents do not submit a bid in each round, or do not follow their complete myopic best-response in each round. For example: can we characterize the problem sets in which *i*Bundle remains allocatively-efficient despite the existence of some "slow" agents in addition to some "fast" agents; can we describe an auction in which agents do not provide their complete exclusive-or bid set in each round, but can provide bids on additional bundles as the auction progresses?

10.2.2 Electronic Commerce Foundations

Multiattribute auctions and combinatorial exchanges present just two new emerging areas of computational mechanism design in electronic commerce.

Multiattribute Auctions

A multiattribute auction allows agents to negotiate over the attributes (size, terms-of-payment, delivery schedule, speed, etc.) of an item in addition to the price. Multiattribute auctions can lead to more efficient outcomes than fixed attribute auctions, in which sellers are restricted to price competition in an *ex ante* fixed space of attributes. Multiattribute auctions promise to provide robust methods for efficient automated negotiation between multiple agents.

There are three central challenges in the design of multiattribute auctions: (1) tractable

winner-determination, which depends on an agent's preferences over the attributes of a good in addition to price; (2) minimize the amount of information revelation required by agents; (3) handle issues of strategic misrepresentation of preferences.

Informationally, it will be important to design iterative mechanisms that allow agents to reveal incremental information about their preferences. Another method to reduce the informational load on users is to position a semi-autonomous proxy bidding agent between a user and the auctioneer that will accept many different types of information, including ordinal, cardinal, hard constraints, functional approximations, and then submit optimal bids to the auctioneer based on this information.

It appears possible to reduce simple multiattribute allocation problems (for example with one seller and multiple buyers) to a combinatorial allocation problem, at least if attributes are discrete, and leverage the *i*Bundle methodology. A bundle of attributes becomes a bundle of items. The preferences of the seller can be introduced via an agent that competes with buyers not to sell goods with particular attributes if the price is too low, or if another set of attributes are more desirable at the current prices. The Myerson-Satterthwaite impossibility theorem limits what can be achieved in a game-theoretic sense. We cannot expect to implement an efficient multiattribute auction that is incentive-compatible for the buyer and the seller *and* budget-balanced, and we must sacrifice one of these conditions. New approaches will be required when the attributes are continuous instead of discrete, and when the solution should allow aggregation across sellers. Both of these changes take the problem further away from the combinatorial auction problem.

Reverse auctions, such as procurement auctions, provide a similar opportunity for *i*Bundle-based methods. In a reverse combinatorial auction there is a single buyer and multiple sellers, each able to provide bundles of items. The efficient allocation will depend on the value of the buyer for different bundles of items and on the costs of each seller to provide bundles. The optimal solution selects bundles of items from multiple sellers to maximize the difference between the value of the buyer and the total cost over the sellers. Again, I believe that it is possible to use a transformation approach, to a regular "forward" combinatorial auction, and derive *i*Bundle price update rules for the reverse auction.

Combinatorial Exchanges

A combinatorial exchange allows multiple buyers to trade with multiple sellers, with all agents able to express logical conditions across bundles of items. Combinatorial exchanges address an important weakness in combinatorial auctions, namely the assumption that there a single seller is able to offer bundles composed of many different types of items. On the contrary, we might expect that a more natural model would allow multiple buyers to engage in combinatorial trades with multiple sellers. Interesting applications of such exchanges include: a bandwidth exchange that aggregates supply from geographically disparate sellers to match bids for "virtual networks", or a travel exchange, that aggregates the supply of excess seats and hotel rooms, to match bids for bundles of rooms and flights. The winner determination problem in a combinatorial exchange, to select bids to maximize surplus, is a classic combinatorial optimization problem.

The pricing problem is interesting, in particular because of the Myerson-Satterthwaite impossibility theorem that shows that it is not possible to achieve efficiency, budget-balance, and incentive-compatibility. We must sacrifice one of these desirable properties. One approach would fix strategy-proofness, and perhaps sacrifice efficiency in favor of budget-balance. This is similar to an approach of McAfee [McA92] for double auctions on homogeneous items. Another approach would fix budget-balance, and try to achieve as much strategy-proofness and efficiency as possible. Parkes, Kalagananam and Eso [PKE01] take this approach, allocating surplus when an exchange is cleared to minimize the distance between agent payments and Vickrey payments. The choice of distance metric has a distributional effect on surplus allocation and changes the incentive properties of the exchange. A simple *threshold rule* appears to perform well, providing partial discounts to agents that would receive a large discount in a Vickrey-Groves scheme.

It remains an open problem to identify special cases in which budget-balance is not a problem in combinatorial exchanges, for example with restricted models of aggregation and bidding languages.

10.2.3 Approximations, Intractability, and Bounded-Rationality

Recent algorithmic approaches to mechanism design consider the effect of approximation and intractability on the economic properties of mechanisms. The goal is to understand both what is possible and what is impossible when tractable computation is introduced as

an additional constraint.

In one sense, bounded-rationality and intractability can help; one might use NP-hardness results to design "bounded strategy-proof" mechanisms that cannot be manipulated without an agent solving a hard problem in polynomial time.

In another sense, bounded-rationality and intractability can be a problem; optimal game-theoretic mechanisms can require that the network infrastructure solves multiple intractable problems, and approximate solutions can quickly break incentive-compatibility properties. In addition to proving worst-case manipulation results for approximation algorithms, one might also use randomized mechanisms to expand the implementation space.

Another thread in this interface between intractability and mechanism design is the effect of agent bounded-rationality on preference revelation. It is often impossible for an agent to compute its complete preference set, i.e. its value for all possible outcomes. The direction started on in this dissertation is to design mechanisms that can solve distributed problems with approximate information revelation from agents, providing dynamic feedback to agents about information collected from other agents and about progress towards a system-wide solution. My current research results suggest that primal-dual optimization theory may provide a suitable set of mathematical tools to develop iterative strategy-proof mechanisms, when used in combination with classic results from mechanism design.

Bibliography

[AS98] Martin Andersson and Tuomas W Sandholm. Leveled commitment contracts with myopic and strategic agents. In *Proc. 15th National Conference on Artificial Intelligence (AAAI-98)*, pages 38–45, July 1998.

[AS99] Martin Andersson and Tuomas W Sandholm. Time-Quality tradeoffs in reallocative negotiation with combinatorial contract types. In *Proc. 16th National Conference on Artificial Intelligence (AAAI-99)*, pages 3–10, July 1999.

[AS00] Martin Andersson and Tuomas W Sandholm. Contract type sequencing for reallocative negotiation. In *Proc. 17th National Conference on Artificial Intelligence (AAAI-00)*, 2000.

[ATY00] Arne Andersson, Mattias Tenhunen, and Fredrik Ygge. Integer programming for auctions with bids for combinations. In *Proc. 4th International Conference on Multi-Agent Systems (ICMAS-00)*, 2000.

[Arr79] Kenneth J Arrow. The property rights doctrine and demand revelation under incomplete information. In M Boskin, editor, *Economics and Human Welfare*. Academic Press, New York, 1979.

[dG79] Claude d'Aspremont and Louis-André Gérard-Varet. Incentives and incomplete information. *J. of Public Economics*, 11:25–45, 1979.

[Aus97] Lawrence M Ausubel. An efficient ascending-bid auction for multiple objects. Technical report, Department of Economics, University of Maryland, 1997.

[AC98] Lawrence M Ausubel and Peter Cramton. The optimality of being efficient. Technical report, University of Maryland, 1998.

[Aus00] Lawrence M Ausubel. An efficient dynamic auction for heterogeneous commodities. Technical report, Department of Economics, University of Maryland, 2000.

[BLP89] Jeffrey S Banks, John O Ledyard, and David Porter. Allocating uncertain and unresponsive resources: An experimental approach. *The Rand Journal of Economics*, 20:1–25, 1989.

[BJ95] Salvador Barberà and Matthew O Jackson. Strategy-proof exchange. *Econometrica*, 63(1):51–87, 1995.

[Ber79] Dimitri P Bertsekas. A distributed algorithm for the assignment problem. Technical report, Lab. for Information and Decision Systems, M.I.T., Cambridge, MA, 1979.

[Ber81] Dimitri P Bertsekas. A new algorithm for the assignment problem. *Math. Progr.*, 21:152–171, 1981.

[Ber86] Dimitri P Bertsekas. Distributed relaxation methods for linear network flow problems. In *Proc. of 25th IEEE Conf. Dec. and Contr.*, pages 2101–2106, 1986.

[Ber87] Dimitri P Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, 1987.

[Ber88] Dimitri P Bertsekas. The auction algorithm: A distributed relaxation method for the assignment problem. *Annals of Operations Research*, 14:105–123, 1988.

[BC89] Dimitri P Bertsekas and David A Castanõn. The auction algorithm for transportation problems. *Annals of Operations Research*, 20:67–96, 1989.

[Ber90] Dimitri P Bertsekas. The auction algorithm for assignment and other network flow problems: A tutorial. *Interfaces*, 20(4):133–149, 1990.

[Bet92] Dimitri P Betsekas. Auction algorithms for network flow problems: A tutorial introduction. *Computational Optimization and Applications*, 1:7–66, 1992.

[BO99] Sushil Bikchandani and Joseph M Ostroy. The package assignment model. Technical report, Anderson Graduate School of Management and Department of Economics, U.C.L.A., 1999.

[BO00] Sushil Bikchandani and Joseph M Ostroy. Ascending price Vickrey auctions. Technical report, Anderson Graduate School of Management, U.C.L.A., 2000.

[BdVSV01] Sushil Bikchandani, Sven de Vries, James Schummer, and Rakesh V Vohra. Linear programming and Vickrey auctions. Technical report, Anderson Graduate School of Management, U.C.L.A., 2001.

[BD89] Mark Boddy and Thomas Dean. Solving time-dependent planning problems. In *Proc. 11th International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 979–984, 1989.

[BGS99] Craig Boutilier, Moises Goldszmidt, and Bikash Sabata. Sequential auctions for the allocation of resources with complementarities. In *Proc. 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 527–534, 1999.

[BP96] Paul J Brewer and Charles R Plott. A binary conflict ascending price (BICAP) mechanism for the decentralized allocation of the right to use railroad tracks. *Int. Journal of Industrial Organization*, 14:857–886, 1996.

[Bre99] Paul J Brewer. Decentralized computation procurement and computational robustness in a smart market. *Economic Theory*, 13:41–92, 1999.

[BCL00] Mark M Bykowsky, Robert J Cull, and John O Ledyard. Mutually destructive bidding: The FCC auction design problem. *J. of Regulatory Economics*, 2000. To appear.

[CL82] Paul Champsaur and Guy Laroque. Strategic behavior in decentralized planning procedures. *Econometrica*, 50:325–344, 1982.

[CR99b] Vijay Chandru and M R Rao. Linear programming. In M.J.Atallah, editor, *Handbook of Algorithms and Theory of Computing*, chapter 31. CRC Press, 1999.

[CR99a] Vijay Chandru and M R Rao. Integer programming. In M.J.Atallah, editor, *Handbook of Algorithms and Theory of Computing*, chapter 32. CRC Press, 1999.

[Che00] Giorgos Cheliotis. Bandwidth trading in the real world: Findings and implications for commodities brokerage. Technical Report RZ 3244, I.B.M. Research Zurich Research Laboratory, 2000.

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

[Cle96] Scott H Clearwater, editor. *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, 1996.

[CS00] Peter Cramton and Jesse Schwartz. Collusive bidding: Lessons from the FCC spectrum auctions. *Journal of Regulatory Economics*, 17, 2000.

[CK81] Vincent P Crawford and E M Knoer. Job matching with heterogeneous firms and workers. *Econometrica*, 49:437–450, 1981.

[DS88] Randall Davis and Reid G Smith. Negotiation as a metaphor for distributed problem solving. In Alan H Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 333–356. Morgan Kaufmann, CA, 1988.

[DGS86] Gabrielle Demange, David Gale, and Marilda Sotomayor. Multi-item auctions. *Journal of Political Economy*, 94(4):863–872, 1986.

[DKLP98] Christine DeMartini, Anthony M Kwasnica, John O Ledyard, and David Porter. A new and improved design for multi-object iterative auctions. Technical Report SSWP 1054, California Institute of Technology, 1998. Revised March 1999.

[DdlVP71] J H Drèze and D de la Vallée Poussin. A tâtonnement process for public goods. *Review of Economic Studies*, 37:133–150, 1971.

[EP94] Carl Ehrman and Michael Peters. Sequential selling mechanisms. *Economic Theory*, 4:237–253, 1994.

[FPS00] Joan Feigenbaum, Christos H Papadimitriou, and Scott Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences, Special Issue on Internet Algorithms*, 2000. To appear. Earlier version in Proc. STOC 2000.

[FNSY96] Donald F. Ferguson, Christos Nikolaou, Jakka Sairamesh, and Yechiam Yemini. Economic models for allocating resources in computer systems. In Clearwater [Cle96], chapter 7, pages 156–183, 1996.

[FT91] Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, 1991.

[FLBS99] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proc. 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 548–553, 1999.

[Gib73] Allan Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41:587–602, 1973.

[Goo71] Irving John Good. Twenty-seven principles of rationality. In V P Godambe and D A Sprott, editors, *Foundations of Statistical Inference*. 1971.

[GSS93] Robert L Graves, Linus Schrage, and Jayaram Sankaran. An auction method for course registration. *Interfaces*, 23(5):81–92, 1993.

[GJJ77] Jerry R Green and Jean-JacquesLaffont. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica*, 45:427–438, 1977.

[GL79] Jerry R Green and Jean-Jacques Laffont. On coailtion incentive compatibility. *Review of Economic Studies*, 46:243–254, 1979.

[GL87] Jerry R Green and Jean-Jacques Laffont. Limited communication and incentive compatibility. In Groves et al. [GRR87], pages 308–329, 1987.

[GIP89] David M Grether, Mark Isaac, and Charles R Plott. *The Allocation of Scarce Resources*. Westview, San Francisco, 1989.

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

[GL77] Theodore Groves and John O Ledyard. Optimal allocation of public goods: A solution to the 'free rider' problem. *Econometrica*, 45:783–810, 1977.

[Gro79] Theodore Groves. Efficient collective choice when compensation is possible. *Review of Economic Studies*, 46:227–241, 1979.

[GRR87] Theodore Groves, Roy Radner, and Stanley Reiter, editors. *Information, Incentives, and Economic Mechanisms.* University of Minnesota Press, 1987.

[GS99] Faruk Gul and Ennio Stacchetti. Walrasian equilibrium with gross substitutes. *Journal of Economic Theory*, pages 95–124, 1999.

[GS00] Faruk Gul and Ennio Stacchetti. The English auction with differentiated commodities. *Journal of Economic Theory*, pages 66–95, 2000.

[Hal93] Susan Fraley Hallowell. *Optimal Dispatching Under Uncertainty: With Application to Railroad Scheduling.* PhD thesis, The Wharton School, University of Pennsylvania, 1993. OPIM TR 93-12-02.

[HTK98] Michael Harkavy, J D Tygar, and Hiroaki Kikuchi. Electronic auctions with private bids. In *Proc. 3rd USENIX Workshop on Electronic Commerce*, 1998.

[Has99] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, pages 627–636, 1999.

[HB00] Holger H Hoos and Craig Boutilier. Solving combinatorial auctions with stochastic local search. In *Proc. 17th National Conference on Artificial Intelligence (AAAI-00)*, pages 22–29, 2000.

[Hor87] Eric J Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *Proc. 3rd AAAI Workshop on Uncertainty in Artificial Intelligence*, pages 429–444, July 1987.

[HC95] Bernardo A Huberman and Scott Clearwater. A multi-agent system for controlling building environments. In *Proc. 1st International Conference on Multi-Agent Systems (ICMAS-95)*, pages 171–176, 1995.

[HG00] Luke Hunsberger and Barbara J Grosz. A combinatorial auction for collaborative planning. In *Proc. 4th International Conference on Multi-Agent Systems (ICMAS-00)*, pages 151–158, 2000.

[Hur72] Leonid Hurwicz. On informationally decentralized systems. In C. McGuire and Roy Radner, editors, *Decision and Organization: A Volume in Honor of Jacob Marchak*. North-Holland, 1972.

[Hur75] Leonid Hurwicz. On the existence of allocation systems whose manipulative Nash equilibria are Pareto optimal. unpublished, 1975.

[HW90] Leonid Hurwicz and Mark Walker. On the generic nonoptimality of dominant-strategy allocation mechanisms: A general theorem that includes pure exchange economices. *Econometrica*, 58:683–704, 1990.

[KL98] Ehud Kalai and John O Ledyard. Repeated implementation. *Journal of Economic Theory*, 83(2):308–317, 1998.

[KC82] Alexander S Kelso and Vincent P Crawford. Job matching, coalition formation, and gross substitutes. *Econometrica*, 50:1483–1504, 1982.

[KG99] Charles D Kolstad and Rolando M Guzman. Information and the divergence between willingness-to-accept and willingness-to-pay. *J Env. Econ. & Manag.*, 38(1):66–80, 1999.

[KHC91] David R Kraay, Patrick T Harker, and B Chen. Optimal pacing of trains in freight railroads: Model formulation and solution. *Operations Research*, 39:82–99, 1991.

[KH95] David R Kraay and Patrick T Harker. Real-time scheduling of freight railroads. *Transportation Research-B*, 29B(3):213–229, 1995.

[KCO97] Per Kreuger, Mats Carlsson, and Jan Olsson. The TUFF train scheduler–trip scheduling on single-track networks. In *CP97 Workshop on Industrial Constraint-Directed Scheduling*, Linz, Austria, 1997.

[KP98] Vijay Krishna and Motty Perry. Efficient mechanism design. Technical report, Pennsylvania State University, 1998.

[Kuh55] Harold W Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[LM82] Jean-Jacques Laffont and Eric Maskin. The theory of incentives: An overview. In W Hildenbrand, editor, *Advances in Economic Theory*, pages 31–94. Cambridge University Press, 1982.

[LS00] Kate Larson and Tuomas W Sandholm. Deliberation in equilibrium: Bargaining in computationally complex problems. In *Proc. 17th National Conference on Artificial Intelligence (AAAI-00)*, pages 48–55, 2000.

[LS01] Kate Larson and Tuomas W Sandholm. Costly valuation computation in auctions. In *Proc. Theoretical Aspects of Rationality and Knowledge VII*, 2001. To appear.

[Led89] John O Ledyard. Incentive compatibility. In John Eatwell, Murray Milgate, and Peter Newman, editors, *Allocation, Information and Markets*, pages 141–151. W. W. Norton, 1989.

[LOP+00] 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 Social Science Working Paper 1093, California Institute of Technology, 2000.

[LPR97] John O Ledyard, David Porter, and Antonio Rangel. Experiments testing multiobject allocation mechanisms. *Journal of Economic and Management Strategy*, 6(3):639–675, 1997.

[Lee85] Tom K Lee. Competition and information acquisition in first price auctions. *Economics Letters*, 18:129–132, 1985.

[LOS99] Daniel Lehmann, Liadan O'Callaghan, and Yoav Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. In *Proc. 1st ACM Conf. on Electronic Commerce (EC-99)*, pages 96–102, 1999.

[Leo83] Herman B Leonard. Elicitation of honest preferences for the assignment of individuals to positions. *Journal of Political Economy*, 91:461–479, 1983.

[LS94] Dan Levin and James L Smith. Equilibrium in auctions with entry. *American Economic Review*, 84:585–599, 1994.

[Mal72] E Malinvaud. Prices for individual consumption, quantity indicators for collective consumption. *Review of Economic Studies*, 39:385–405, 1972.

[Mar87] Thomas Marschak. Private versus direct revelation: Informational judgements for finite mechanisms. In Groves et al. [GRR87], pages 132–, 1987.

[MR72] Jacob Marschak and Roy Radner. *Economic theory of teams.* Yale University Press, New Haven, 1972.

[MCWG95] Andreu Mas-Colell, Michael D Whinston, and Jerry R Green. *Microeconomic Theory.* Oxford University Press, 1995.

[Mat84] Steven A Matthews. Information acquisition in discriminatory auctions. In M Boyer and R Kihlstrom, editors, *Bayesian Models in Economic Theory.* North Holland, 1984.

[PMM87] R Preston McAfee and John McMillan. Auctions and bidding. *Journal of Economic Literature*, 25:699–738, June 1987.

[McA92] R Preston McAfee. A dominant strategy double auction. *J. of Economic Theory*, 56:434–450, 1992.

[MM96] R Preston McAfee and John McMillan. Analyzing the airwaves auction. *J Econ Perspect*, 10:159–175, 1996.

[McM94] John McMillan. Selling spectrum rights. *Journal of Economic Perspectives*, 8:145–62, 1994.

[Mil00a] Paul Milgrom. Putting auction theory to work: Ascending auctions with package bidding. Technical report, Stanford and MIT, 2000.

[Mil00b] Paul Milgrom. Putting auction theory to work: The simultaneous ascending auction. *Journal of Political Economy*, 108:245–272, 2000.

[MW82] Paul Milgrom and Robert Weber. A theory of auctions and competitive bidding. *Econometrica*, 50:1089–1122, 1982.

[MR88] John Moore and Rafael Repullo. Subgame perfect implementation. *Econometrica*, 56(5):1191–1220, September 1988.

[MR74] Kevin Mount and Stanley Reiter. The informational size of message spaces. *Journal of Economic Theory*, 8:161–191, 1974.

[MS99] Piero La Mura and Yoav Shoham. Expected utility networks. In *Proc. 15th Conference on Uncertainty in Artificial Intelligence*, pages 366–373, 1999.

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

[Mye81] Roger B Myerson. Optimal auction design. *Mathematics of Operation Research*, 6:58–73, 1981.

[Mye83] Roger B Myerson. Mechanism design by an informed principal. *Econometrica*, 51:1767–1797, 1983.

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner Privacy preserving auctions and mechanism design. In *Proc. 1st ACM Conf. on Electronic Commerce (EC-99)*, pages 129–139, 1999.

[Nas50] John Nash. Equilibrium points in n-person games. In *Proceedings of the National Academy of Sciences*, volume 36, pages 48–49, 1950.

[Nis00] Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proc. 2nd ACM Conf. on Electronic Commerce (EC-00)*, pages 1–12, 2000.

[NR00] Noam Nisan and Amir Ronen. Computationally feasible VCG mechanisms. In *Proc. 2nd ACM Conf. on Electronic Commerce (EC-00)*, pages 242–252, 2000.

[NR01] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

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

[PS82] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, 1982.

[PUF99] David C Parkes, Lyle H Ungar, and Dean P Foster. Accounting for cognitive costs in on-line auction design. In Pablo Noriega and Carles Sierra, editors, *Agent Mediated Electronic Commerce (LNAI 1571)*, pages 25–40. Springer-Verlag, 1999. Earlier version appeared at the Agents'98 Workshop on Agent Mediated Electronic Trading, 1998.

[Par99] David C Parkes. *i*Bundle: An efficient ascending price bundle auction. In *Proc. 1st ACM Conf. on Electronic Commerce (EC-99)*, pages 148–157, 1999.

[PU00a] David C Parkes and Lyle H Ungar. Iterative combinatorial auctions: Theory and practice. In *Proc. 17th National Conference on Artificial Intelligence (AAAI-00)*, pages 74–81, 2000.

[PU00b] David C Parkes and Lyle H Ungar. Preventing strategic manipulation in iterative auctions: Proxy agents and price-adjustment. In *Proc. 17th National Conference on Artificial Intelligence (AAAI-00)*, pages 82–89, 2000.

[Par01] David C Parkes. An iterative generalized Vickrey auction: Strategy-proofness without complete revelation. In *Proc. AAAI Spring Symposium on Game Theoretic and Decision Theoretic Agents.* AAAI Press, March 2001.

[PU01] David C Parkes and Lyle H Ungar. An auction-based method for decentralized train scheduling. In *Proc. 5th International Conference on Autonomous Agents (AGENTS-01)*, 2001. To appear.

[PKE01] David C Parkes, Jayant Kalagnanam, and Marta Eso. Vickrey-based surplus distribution in combinatorial exchanges. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, 2001. To appear.

[Plo97] Charles R Plott. Laboratory experimental testbeds: Application to the PCS auction. *Journal of Economics and Management Strategy*, 6(3):605–638, 1997.

[Por99] David P Porter. The effect of bid withdrawal in a multi-object auction. *Review of Economic Design*, pages 73–97, 1999.

[Rad87] Roy Radner. Decentralization and incentives. In Groves et al. [GRR87], pages 3–47, 1987.

[Rad92] Roy Radner. Hierarchy: The economics of managing. *Journal of Economic Literature*, 30:1382–1415, 1992.

[Rad93] Roy Radner. The organization of decentralized information processing. *Econometrica*, 61:1109–1146, 1993.

[RSB82] 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.

[Rei74] Stanley Reiter. Informational efficiency of iterative processes and the size of message spaces. *Journal of economic theory*, 8:389–396, 1974.

[Rob87] John Roberts. Incentives, information and iterative planning. In Groves et al. [GRR87], chapter 13, pages 349–374, 1987.

[RP76] John Roberts and Andrew Postlewaite. The incentives for price-taking behavior in large exchange economies. *Econometrica*, 44:115–128, 1976.

[Rob79] Kevin Roberts. The characterization of implementable rules. In Jean-Jacques Laffont, editor, *Aggregation and Revelation of Preferences*, pages 321–348. North-Holland, Amsterdam, 1979.

[RZ94] Jeffrey S Rosenschein and Gilad Zlotkin. *Rules of Encounter*. MIT Press, 1994.

[RPH98] Michael H Rothkopf, Aleksandar Pekeč, and Ronald M Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[RSP93] Stuart Russell, Devika Subramanian, and Ronald Parr. Provably bounded optimal agents. In *Proc. 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 338–344, 1993.

[Rus95] Stuart Russell. Ratonality and intelligence. In *Proc. 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 950–957, August 1995.

[RW91] Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial Intelligence*, 49:361–395, 1991.

[Sam85] William F Samuelson. Competitive bidding with entry costs. *Economic Letters*, 17:53–57, 1985.

[San93] Tuomas W Sandholm. An implementation of the Contract Net Protocol based on marginal-cost calculations. In *Proc. 11th National Conference on Artificial Intelligence (AAAI-93)*, pages 256–262, July 1993.

[SL95] Tuomas W Sandholm and Victor R Lesser. Issues in automated negotiation and electronic commerce: Extending the Contract Net framework. In *Proc. 1st International Conference on Multi-Agent Systems (ICMAS-95)*, pages 328–335, 1995.

[SL96] Tuomas W Sandholm and Victor R Lesser. Advantages of a leveled commitment contracting protocol. In *Proc. 13th National Conference on Artificial Intelligence (AAAI-96)*, pages 126–133, July 1996.

[San96] Tuomas W Sandholm. Limitations of the Vickrey auction in computational multiagent systems. In *Second International Conference on Multiagent Systems (ICMAS-96)*, pages 299–306, 1996.

[SL97] Tuomas W Sandholm and Victor R Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1–2):99–137, 1997.

[San99] Tuomas W Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Proc. 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 542–547, 1999.

[San00] Tuomas W Sandholm. Issues in computational Vickrey auctions. *International Journal of Electronic Commerce*, 4:107–129, 2000.

[San94] Jayaram K Sankaran. On a dynamic auction mechanism for a bilateral assignment problem. *Mathematical Social Sciences*, 28:143–150, 1994.

[Sat75] Mark A Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.

[Sch97] James Schummer. Strategy-proofness vs. efficiency on restricted domains of exchange economies. *Social Choice and Welfare*, 14:47–56, 1997.

[ST01] Yoav Shoham and Moshe Tennenholtz. On rational computability and communication complexity. *Games and Economic Behavior*, 35:197–211, 2001.

[SBG94] Avraham Shtub, Jonathan F Bard, and Shlomo Globerson. *Project Management: Engineering, Technology and Implementation.* Prentice-Hall, Inc., 1994.

[Sim76] Herbert A Simon. From substantive to procedural rationality. In S J Latsis, editor, *Method and Appraisal in Economics*, pages 129–148. Cambridge University Press, 1976.

[Ste96] Mark Stegeman. Participation costs and efficient auctions. *Journal of Economic Theory*, 71:228–259, 1996.

[SDK+94] Michael Stonebraker, Robert Devine, Marcel Kornacker, Witold Litwin, Avi Pfeffer, Adam Sah, and Carl Staelin. An economic paradigm for query processing and data migration in Mariposa. In *Proc. 3rd Int. Conf. on Parallel and Distributed Information Systems*, pages 58–67, 1994.

[TKDM00] Moshe Tennenholtz, Noa Kfir-Dahav, and Dov Monderer. Mechanism design for resource bounded agents. In *Proc. 4th International Conference on Multi-Agent Systems (ICMAS-00)*, 2000.

[TW81] Jorgen Tind and Laurence A Wolsey. An elementary survey of general duality theory in mathematical programming. *Mathematical Programming*, 21:241–261, 1981.

[Var95] Hal R Varian. Economic mechanism design for computerized agents. In *Proc. USENIX Workshop on Electronic Commerce*, 1995. Minor update, 2000.

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

[vNM47] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, second edition, 1947.

[dVV00] Sven de Vries and Rakesh V Vohra. Combinatorial auctions: A brief survey. Technical report, MEDS Department, Kellogg Graduate School of Management, Northwestern University, 2000.

[WWY00] William E Walsh, Michael P Wellman, and Fredrik Ygge. Combinatorial auctions for supply chain formation. In *Proc. ACM Conference on Electronic Commerce*, pages 260–269, 2000.

[Wel93] Michael P Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

[Wel96] Michael P Wellman. Market-oriented programming: Some early lessons. In Clearwater [Cle96], chapter 4, pages 74–95, 1996.

[WWWMM01] Michael P Wellman, William E Walsh, Peter R Wurman, and Jeff K MacKie-Mason. Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 35:271–303, 2001.

[WWO+01] Michael P Wellman, Peter R Wurman, Kevin O'Malley, Roshan Bangera, Shou-de Lin, Daniel M Reeves, and William E Walsh. Designing the market game for a trading agent competition. *IEEE Internet Computing*, pages 43–51, 2001.

[Wil99] Steven R Williams. A characterization of efficient, Bayesian incentive-compatible mechanisms. *Economic Theory*, 14:155–180, 1999.

[Wol81a] Laurence A Wolsey. Integer programming duality: Price functions and sensitivity analysis. *Mathematical Progamming*, 20:173–195, 1981.

[Wol81b] Laurence A Wolsey. A resource decomposition algorithm for general mathematical programs. *Mathematical Programming Study*, 14:244–257, 1981.

[Wur99] Peter R Wurman. *Market Structure and Multidimensional Auction Design for Computational Economies*. PhD thesis, University of Michigan, 1999.

[WW99] Peter R Wurman and Michael P Wellman. Equilibrium Prices in Bundle Auctions. In *AAAI-99 Workshop on Artificial Intelligence for Electronic Commerce*, pages 56–61, 1999.

[WWW00] Peter R Wurman and Michael P Wellman and William E Walsh. A parameterization of the auction design space. *Games and Economic Behavior*, 35:304–338, 2001.

[WW00] Peter R Wurman and Michael P Wellman. A*k*BA: A progressive, anonymous-price combinatorial auction. In *Second ACM Conference on Electronic Commerce*, pages 21–29, 2000.