

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 $i = 1, \dots, I$, 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*, $f : (\Theta_1, \times, \dots, \times, \Theta_I) \rightarrow \mathcal{O}$, that selects an optimal outcome $o^* = f(\theta)$ based on the types $\theta = (\theta_1, \dots, \theta_I)$ of all agents.

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

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

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.

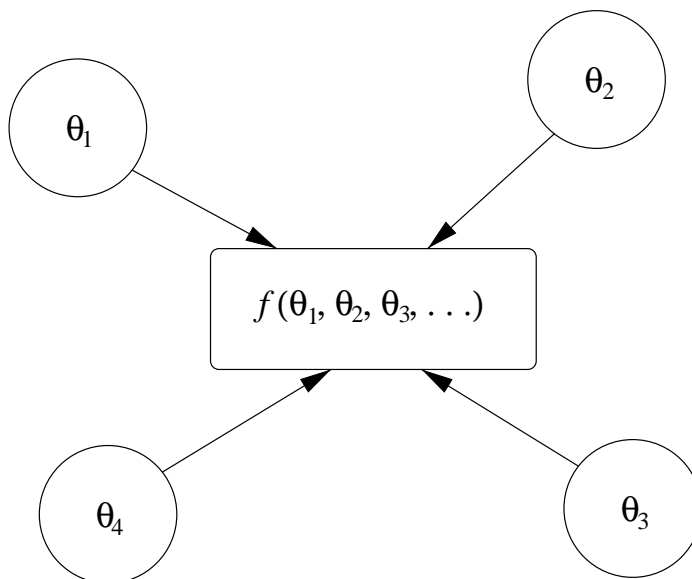


Figure 1.1: The implementation problem.

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 independent 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, *iBundle 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 $v_i : 2^{\mathcal{G}} \rightarrow \mathbb{R}_+$, 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 $\mathbf{S} = (S_1, \dots, S_I)$ 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.

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

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

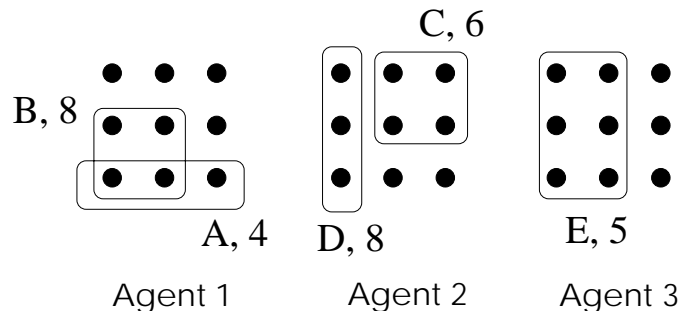


Figure 1.2: An example combinatorial allocation problem.

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 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 *iBundle 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. *iBundle* 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

¹This 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 \mathcal{B} Bundle remains NP-hard. However, each problem instance in \mathcal{B} 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 \mathcal{B} 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 \mathcal{B} 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 \mathcal{B} 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, A*k*BA [Wur99, WW00]. The main differences between A*k*BA and *i*Bundle are in the structure of prices and price updates. A*k*BA 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 *iBundle* 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 *iBundle*, 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 *iBundle* 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 *iBundle*.

Chapter 7 presents the Extend&Adjust method, which introduces a second phase to *iBundle*. 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 *iBundle 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, *iBundle* 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.