RESOURCE ALLOCATION IN COMPETITIVE MULTIAGENT SYSTEMS

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

Kevin Leyton-Brown
August 2003

Abstract

In systems involving multiple autonomous agents, it is often necessary to decide how scarce resources should be allocated. When agents have competing interests, they may have incentive to deviate from protocols or to lie to other agents about their preferences. Due to the strategic nature of such interactions, there has been a recent surge of interest in addressing problems in competitive multiagent systems by bringing together techniques from computer science and game-theoretic economics. In some cases, the interesting issue is the application of ideas from computer science to make existing economic mechanisms practical. In other cases, selfish agents' conflicting demands of a computer system can best be understood and/or managed through game-theoretic analysis. This thesis addresses problems that fall into both cases.

The first part of this work considers game-theoretic issues in multiagent resource allocation:

- using incentives to diffuse temporally-focused usage of a resource on a computer network at the lowest possible cost;
- identifying a protocol to allow agents to cooperate in (single-good) first-price auctions, and showing that in equilibrium such collusion benefits both colluding and non-colluding agents at the auctioneer's expense;
- compactly representing strategic multiagent situations as local-effect games, a novel class of multi-player general-sum games for which pure-strategy Nash equilibria can often be proven to exist and be computed efficiently.

The second part considers computational issues in multiagent resource allocation, concentrating on the winner determination problem (WDP) in combinatorial auctions:

- solving several variants of the WDP using heuristic branch-and-bound search algorithms;
- building a benchmark suite for WDP algorithms by modeling real-world valuations described in the economics literature;
- modeling the (empirical) computational hardness of the WDP, analyzing these models, and applying them to construct algorithm portfolios and to generate harder problem instances.

Acknowledgements

Academic influences

Foremost, Yoav Shoham has been an ideal advisor for me. He gave me the freedom to develop in my own direction while still providing strong guidance, making sure that I focused on the most interesting problems and didn't neglect fruitful avenues. Yoav has been generous in every way, providing me with introductions, invitations to academic, industry and government meetings, the means to attend a wide variety of conferences all around the world, the freedom to build a large computer cluster, and of course innumerable hours of his own time. My research meetings with Yoav are always exhausting: in five minutes he can identify a weak point in an argument that I have accepted for a month. It is an understatement to say that Yoav has taught me a great deal.

Moshe Tennenholtz has all but co-advised me; during his three years as a visiting scholar at Stanford he had a profound impact on my development as a researcher. Our hours-long conversations spent attacking a problem have been among the most rewarding I've had in research. His insight, his attention to detail and his knowledge of the literature all served to carry our work much farther than would have been possible otherwise. On a more personal note, Moshe is a very open, giving person, and I have benefited many times from his kindness.

Many other people at Stanford have taught, guided and helped me. Balaji Prabhakar mentored me on issues in networking; without his involvement the work in Chapter 2 could never have even gotten off the ground. Daphne Koller is an amazing academic, and has challenged and inspired me throughout my time at Stanford.

Andrew Ng gave very generously of his time, making extensive (and very helpful) comments on a draft of this thesis. The many students with whom I worked at Stanford taught me a lot, and also made doing research much more fun. Particularly, I'd like to thank Eugene Nudelman and Ryan Porter, my officemates for most of my time at Stanford, for many useful (and diverting) discussions. Finally, I'd like to thank the Stanford professors who helped me by serving on various committees as I progressed through my Ph.D: the members of my reading committee (Andrew Ng, Yoav Shoham, Moshe Tennenholtz), my orals committee (reading committee plus Daphne Koller, David Kreps) and my quals committee (Daphne Koller, Chris Manning, Moshe Tennenholtz).

There are of course *many* people in the larger research community who have influenced me academically. Though this list is certainly incomplete, I'd particularly like to thank Craig Boutilier, Peter Cramton, Carla Gomes, Amy Greenwald, David Parkes, Tuomas Sandholm, Bart Selman, Bill Walsh, Mike Wellman and Peter Wurman.

Coauthors

Over the past five years I have been lucky enough to work with outstanding coauthors. I am indebted to them for their many contributions to the work described in this thesis.

Chapter 2 is based on joint work with Ryan Porter, Balaji Prabhakar, Yoav Shoham and Shobha Venkataraman. An early extended abstract was [Leyton-Brown *et al.*, 2001]; the journal version was [Leyton-Brown *et al.*, 2003c].

Chapter 3 is based on joint work with Yoav Shoham, Moshe Tennenholtz and Navin Bhat. Thanks to Ryan Porter and Yossi Feinberg for very helpful discussions. This work received additional financial support from an Ontario Graduate Scholarship. An early version was [Leyton-Brown *et al.*, 2000c]; [Leyton-Brown *et al.*, 2002b] was a more recent extended abstract.

Chapter 4 is based on joint work with Moshe Tennenholtz. The conference version was [Leyton-Brown & Tennenholtz, 2003].

Chapter 6 is based on joint work with Yuzo Fujishima, Yoav Shoham and Moshe Tennenholtz. The CASS algorithm was first presented in [Fujishima *et al.*, 1999]; the CAMUS algorithm was introduced in [Leyton-Brown *et al.*, 2000b].

Chapter 7 is based on joint work with Mark Pearson and Yoav Shoham. The conference version was [Leyton-Brown *et al.*, 2000a].

Chapter 8 is based partly on unpublished data collected by Eugene Nudelman.

Chapter 9 is based on joint work with Eugene Nudelman and Yoav Shoham. This work received additional financial support from the Intelligent Information Systems Institute (IISI), Cornell. Thanks to Rámon Béjar, Carla Gomes, Henry Kautz, Bart Selman, Lyle Ungar and Ioannis Vetsikas for helpful discussions. A conference version was [Leyton-Brown *et al.*, 2002a].

Chapter 10 is based on joint work with Eugene Nudelman, Galen Andrew, Jim McFadden and Yoav Shoham. This work also received support from IISI. Two extended abstracts describing this work were [Leyton-Brown *et al.*, 2003b] and [Leyton-Brown *et al.*, 2003a].

The work in this thesis was funded by DARPA grant F30602-00-2-0598, a Stanford Graduate Fellowship, and a Canadian Natural Sciences and Engineering Research Council PGS-A fellowship.

Others who offered me support

First, my extended family (Mom, Dad, Mlle, Allison and Caleigh, and enthusiastic grandparents, uncles and aunt) have been cheering me on every step of the way. They have set an example by the value they place on learning and the questioning of assumptions and by their integrity, both intellectual and otherwise. I hope I am able to live up to this tradition in my academic life and beyond.

My girlfriend Judith has helped, supported and influenced me in more ways than I can list here. I'd like to thank her particularly for the many sacrifices she made to live with me in California while I worked on my Ph.D, and also for tolerating the havoc that my research obligations played with our schedule. Her love and support have helped me through several tough times during the past four years, and more than anything else have defined my time at Stanford.

Finally, I'd like to thank my friends, with the caveat that there are many people that space does not permit me to mention here. Carlos and Philippe are friends I met during the Stanford recruitment weekend in the summer of 1998. We have been together for just about everything over the past five years: hiking trips, impromptu tutorials on each others' research areas, dot-com fantasies, international travel, chats in our offices while avoiding work, and ultimately seeking work in our respective job hunts. Geoff was around for a lot of the above, and also introduced me to the appreciation of wine. I've also been lucky with randomly-assigned roommates; in particular, Dan is a great friend who introduced me to many new people and experiences; Farhad and Irfan gave me a welcoming introduction to Pakistani culture and food; and Ryan made my final year (separated from Jude while she finished her Master's in New Zealand) much more lively and enjoyable.

I'm lucky to have remained in close contact with many friends back home in Canada, particularly Christa, Tara, Jai and Navin. Navin deserves particular mention here as he found time during his own (physics) Ph.D. studies to join me as a coauthor on the work described in Chapter 3.

I'd like to thank the members of the DAGS and multiagent research groups. Many DAGS members have been both friends and informal teachers: I've been lucky to have them around for everything from ski trips to qual study groups. (I'd particularly like to thank Ben for explaining SVMs to me on a ski lift—in between black-diamond mogul runs!) Finally, it often seems that I've learned more from the members of the multiagent group than anywhere else in my time at Stanford. I thus have a long list of people to thank (many of whom also featured in the list of coauthors): Alex, Bikash, Bob McGrew, Bob Wilson, Christian, Eugene, Galen, Jenn, Jim, Karèn, Mark, Moises, Moshe, Pedrito, Piero, Rob, Ryan, Shobha and of course Yoav. I've always found that this lively group has struck the perfect balance between being friendly and accepting on one hand, and pushing me to learn more and to think more clearly on the other. No longer having so many of these people just a desk or an office away will be one of the hardest things about leaving Stanford.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Multiagent Systems

In recent years, and especially with the advent of the internet, there has been increasing interest in multiagent systems. Different researchers do not always agree about what sort of systems to consider "multiagent," however; in particular there are differences of opinion about where to draw a line between "agents" and "objects."[1] I'll begin by stating the characteristics that I see as fundamental to multiagent systems:

1. Agents are autonomous;

2. Information is distributed asymmetrically, and agents must choose what information to share with others (and in some cases whether to tell the truth).

Agents must therefore make autonomous decisions about how to act, taking into account their own local information and their knowledge about the presence, desires and declared local information of other agents.

Multiagent systems can broadly be divided into two categories. *Cooperative multiagent systems* are those in which all agents share the same desires. Therefore, given complete information about the world there is some outcome or set of outcomes which

[1]That is, encapsulated blocks of data and code common to object-oriented programming languages such as C++ and Java.

every agent agrees is preferable to all other outcomes. Of course these outcomes may be difficult to achieve in practice, since physical separation, insufficient communication bandwidth or computational limitations may prevent agents from achieving common knowledge of all relevant information or from determining which outcomes they prefer given this information. An example of a cooperative multiagent system is a team of robots which act together to achieve an objective in an unknown environment.

A more general class is *competitive multiagent systems*. Here we relax the assumption that all agents have the same interests, assuming only that each agent's preferences can be described by some utility function. In a sense, the term "competitive" is misleading, since some utility functions can lead agents to want to cooperate with each other, or to cooperate in some situations and to compete in others. The key way in which competitive multiagent systems differ from the cooperative variety is that agents cannot be *assumed* to have the same interests: even in situations where agents are best off cooperating they may not realize it, or may not behave as though their interests are aligned. An example of a competitive multiagent system is a peer-to-peer file sharing system, in which not all agents affect each other, but in which agents compete for scarce bandwidth and attempt to avoid having to provide files for other agents.

1.2 Resource Allocation in Multiagent Systems

The above example illustrates a common feature of multiagent systems: the presence of *resources*, scarce commodities of which agents make use and which the system explicitly or implicitly allocates. Indeed, many multiagent systems exist primarily to distribute such resources. In cooperative multiagent systems it is comparatively simple to allocate resources optimally. A global policy can be established according to which agents first communicate relevant information and the system then determines an allocation. This allocation can be computed by a central authority, by a distinguished agent, or implicitly by a group of agents. Because all agents prefer the same outcome(s) given full information, the notion of an optimal allocation is well-defined.

Things are not so clear in the case of competitive multiagent systems. First of all, individual agents may not have incentive to declare their private information in order to allow the system to select its chosen outcome. Instead, they may be able to lie about their information in a way that will cause the system to select an outcome which they prefer. Protocols must therefore be defined to give selfish agents incentive to truthfully declare any relevant private information, by ensuring that agents hurt themselves by lying. Secondly, there is no unequivocal notion of optimality in a competitive multiagent system. Instead, systems can be designed to achieve one of a number of different—and often mutually exclusive—objectives: social-welfare maximization (*i.e.*, maximizing the sum of agents' utilities), revenue maximization (*i.e.*, maximizing the amount that agents are made to pay in exchange for the resources allocated to them), pareto-optimality (finding an outcome satisfying the condition that there does not exist another outcome in which no agent is worse off and some agent is better off), and so on.

The past few years have seen a surge of interest in addressing these problems by bringing together techniques from computer science and game theoretic economics. After all, microeconomics has been concerned with designing protocols for the allocation of scarce resources among selfish agents for half a century. Computer science and economics have made contact in several different ways. In some cases, the interesting problem is to apply ideas from computer science to make it practical to use existing economic mechanisms. In other cases, selfish agents' conflicting demands of a computer system can best be managed by the introduction of game theoretic incentives. The influence between the disciplines can also be more symmetric, requiring a synthesized view of both computational limitations and the self-interested behavior of agents.

1.3 Overview

This thesis considers problems that fall into all of these categories. The topics covered are divided into two parts, based on whether the work focuses on game-theoretic or computational issues in multiagent resource allocation. In Figure 1.1 these topics are

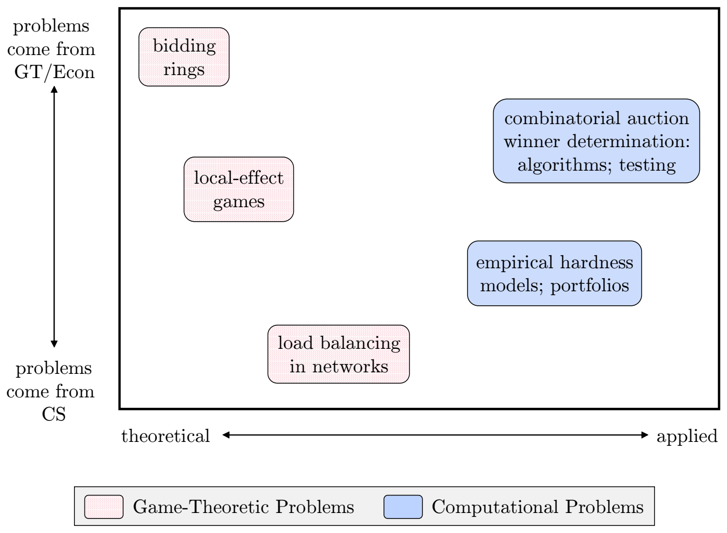

problems come from GT/Econ
problems come from CS
theoretical
applied
bidding rings
local-effect games
load balancing in networks
combinatorial auction winner determination: algorithms; testing
empirical hardness models; portfolios
Game-Theoretic Problems
Computational Problems

Figure 1.1: Topics covered in this thesis

related to each other in terms of whether they originate in computer science or in game-theoretic economics, and of the extent to which the approach used in addressing them is either theoretical or applied.

1.3.1 Game-Theoretic Topics

In this part, we examine three different game-theoretic issues in competitive multiagent systems: the application of game-theoretic incentives to computer networks; cooperation among selfish agents in an auction; and the computation of pure-strategy Nash equilibria in compactly-represented games.

Chapter 2 investigates an incentive-based approach to balancing the load on a network resource. Traditionally, network loads are managed using a technological approach (*e.g.*, fair-queueing systems); however, when loads result from agents' selfish actions, better solutions can be achieved at a lower cost when new incentives are provided. More specifically, we consider the problem of sharing network resources when users' preferences lead to temporally concentrated loads, resulting in inefficient use of the network. We consider a setting in which bandwidth or access to service is available during different time slots at a fixed cost, but all agents have a natural preference for choosing the same time slot. We present four mechanisms that motivate users to distribute the load by probabilistically waiving the cost for each time slot, and analyze the equilibria that arise under these mechanisms.

Chapter 3 considers the problem of designing a protocol (a “bidding ring”) to allow agents to cooperate in a first-price sealed-bid auction, so that at least one of the cooperating agents benefits and no agent has an incentive to deviate from the protocol. Unlike previous work on this topic, the work presented in this chapter allows for the existence of multiple cartels in the auction, includes the choice of whether or not to collude as part of agents’ strategy space, and does not assume that non-colluding agents hold false beliefs. We show a Bayes-Nash equilibrium in which agents choose to join bidding rings when invited and to truthfully declare their valuations to a ring center. Furthermore, we show that the existence of bidding rings benefits ring centers and all agents, both members and non-members of bidding rings, at the auctioneer’s expense.

Finally, Chapter 4 considers the problem of compactly representing realistic games, and leveraging this compact representation to yield tractable computation of Nash equilibria. Many of the real-world multiagent systems for which game-theoretic representations are natural involve many players and many actions for each player; however, current algorithmic techniques for the computation of Nash equilibria are unable to scale up beyond much smaller games. We introduce a new class of games, local-effect games (LEGs), which model several realistic settings and also admit compact representation. We show both theoretically and empirically that these games often (but not always) have pure-strategy Nash equilibria. Finding a potential function is a good technique for finding such equilibria. We give necessary and sufficient conditions for LEGs to have potential functions and provide the functions for the cases in which they exist; we also show a general case where pure-strategy equilibria exist in the absence of potential functions. In experiments, we show that myopic best-response dynamics converge quickly to pure strategy equilibria in a class of games not covered by our positive theoretical results.

1.3.2 Computational Topics

The second part of this thesis investigates a computational problem in competitive multiagent systems. In the past half-decade computer scientists have devoted a great deal of study to combinatorial auctions (CAs), a resource-allocation paradigm which allows agents to indicate their interest in arbitrary bundles of goods even when their valuations for these goods are non-additive. The inherent flexibility of combinatorial auctions makes them useful for a wide variety of resource allocation applications ranging from process scheduling on shared computational resources to the federal government's multi-billion-dollar sale of radio spectrum licenses. However, combinatorial auctions have an Achilles' heel: they require the use of a complex and computationally expensive optimization procedure to identify the set of winning bids. We examine this combinatorial auction winner determination problem (WDP), studying the design of WDP algorithms, WDP benchmark distributions, the use of machine learning techniques to build models of the empirical running time of such algorithms on such benchmark data, and various applications of these empirical hardness models.

We begin with an introduction to combinatorial auctions and various formulations of the winner determination problem in Chapter 5. Chapter 6 considers the design of algorithms for the WDP. Two algorithms are presented, CASS and CAMUS. The former is a branch-and-bound search algorithm, which is guaranteed to find an optimal solution and which uses a variety of heuristics to perform well in practice. CAMUS is a generalization of CASS which handles an important variant of combinatorial auctions in which each good at auction may have multiple identical "units."

Chapter 7 describes a benchmark suite which has been widely used for the evaluation of WDP algorithms. Since very few general CA's have been held, there is little data recording the bidding behavior of real bidders. Instead, to evaluate WDP algorithms we must generate artificial data that is as representative of realistic scenarios as possible. We present five instance generators, each modeling a real-world application of combinatorial auctions described in the economics literature.

Chapter 8 provides experimental results for the CASS algorithm on the benchmark suite presented in Chapter 7. We demonstrate CASS's scaling and anytime properties and then compare CASS to the two most widely-cited approaches to solving the WDP:

Sandholm's Bidtree algorithm and ILOG's CPLEX integer programming solver.

Chapter 9 investigates the empirical hardness of the winner determination problem. We identify a large number of distribution-nonspecific features of data instances and use statistical regression techniques to learn, evaluate and interpret a function from these features to the predicted hardness of an instance, focusing mostly on ILOG's CPLEX solver. Surprisingly, we find that it is possible to build very accurate models of running time, even though this value varies substantially on problem instances involving the same number of bids and goods.

Finally, Chapter 10 presents two applications of our empirical hardness models. First, we show how to build algorithm portfolios that select among a set of black-box algorithms, and give experimental results using CPLEX, CASS and one other special-purpose WDP algorithm. Second, we present techniques for using empirical hardness models to make benchmark distributions harder, and evaluate these techniques on the distributions introduced in Chapter 7.

Part I

Game-Theoretic Issues in Multiagent Resource Allocation

Chapter 2

Congestion Management in Networks

We explore the problem of sharing network resources when users' preferences lead to temporally concentrated loads, resulting in an inefficient use of the network. In such cases external incentives can be supplied to smooth out demand, obviating the need for expensive technological mechanisms. Taking a game-theoretic approach, we consider a setting in which bandwidth or access to service is available during different time slots at a fixed cost, but all agents have a natural preference for choosing the same time slot. We present four mechanisms that motivate users to distribute the load by probabilistically waiving the cost for each time slot, and analyze the equilibria that arise under these mechanisms.

2.1 Introduction

Competition for network resources is intrinsic to a network's operation and leads to congestion. Since users access resources in a distributed and uncoordinated fashion, it is common for a network to experience congestion even when the average demand for a resource is much less than its capacity. Some of these congestion epochs are simply a product of the statistical nature of user access patterns and traffic types, and are thus unpredictable. To cope with this lack of coordination among users and the unpredictability of congestion epochs, networks send “congestion signals” to users to help them share its resources in a fair and satisfactory fashion. For example, packets at a congested router may be either dropped or marked [Floyd, 1994].

A great deal of network congestion is not only caused by a lack of coordination, but also by users who aim to selfishly maximize the bandwidth available to them (see Shenker [1995]). There exists a substantial body of work on the fair management of this sort of congestion in networks. In particular, the problem of designing congestion control and pricing mechanisms to provide differentiated qualities-of-service (QoS) in the Internet has received a lot of attention recently. The first common type of solution to this problem is technological: the network can erect “bandwidth firewalls” between packet flows using scheduling algorithms like Weighted Fair Queuing [Demers *et al.*, 1990]. Such scheduling algorithms decrease or eliminate the dependence of one flow’s QoS from the QoS of other flows. They can be difficult to implement in high-capacity routers, however, as they require the maintenance of per-flow state to distinguish, buffer and schedule the packets of individual flows. This has led researchers to explore trading off performance for simplicity of implementation, yielding router mechanisms that provide approximate fairness [Floyd & Jacobson, 1993; Pan *et al.*, 2000; Pan *et al.*, 2001].

An alternate line of research takes an economic approach to congestion management. Following this approach the network attempts to induce users to condition their flows; this avoids the implementation complexity inherent in erecting explicit bandwidth firewalls. Using ideas from economics, MacKie-Mason and Varian [1994] argued that this incentive can be provided by charging agents for the damage caused to others by their ill-conditioned flows. This work proposes a “smart market” that uses bids to set a price for network usage at each of several time slots. Gibbens and Kelly [1999] suggest charging a user for the role its packets play in causing congestion; see also [Gibbens & Key, 1999] and [Key & McAuley, 1999]. Odlyzko [1997] proposes “Paris Metro Pricing”: partitions of the network that behave identically but charge different prices, inviting users to choose the partition they believe will offer the best balance of cost and congestion.

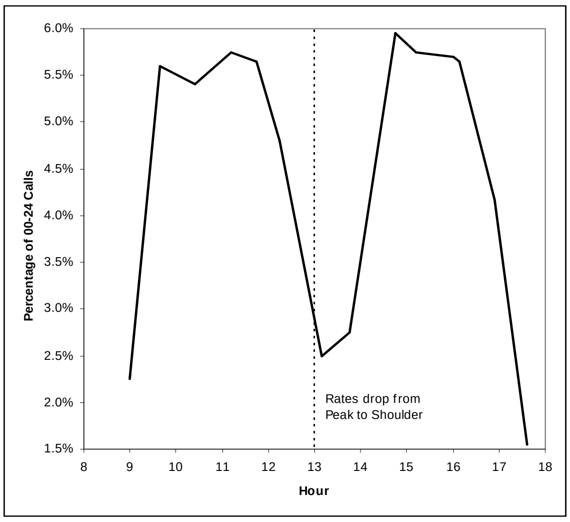


Figure 2.1: Quarterly Trunk Calls on Weekdays in the United Kingdom, December 1975

In some situations, times of high demand are regular and predictable. Such *focused loading* can occur because many users' utility functions are maximized by using the network at some specific time. For example, early studies of long-distance telephone networks show a spike in usage when rates drop [Mitchell, 1978]. Figure 2.1 is a representative graph adapted from p. 450 of this paper, which shows telephone network traffic versus time of day. Note that usage falls off before the 1 PM rate drop, spikes afterwards and then falls off again. A recent study [Blair, 1998] considers dial-up data traffic in Ireland and the UK— where ISPs provide free Internet access but users pay for the duration of their phone connections—showing that a focused load on the telephone network occurs due to an increase in data connections when phone charges drop. Web servers also experience focused loading just before deadlines, or just after new content or services are made available. While these times are known well in advance, users have no incentive to avoid accessing the web site close to the deadline and thus can cause server overloads or crashes, to which system managers typically respond by buying more resources.

What approaches would more directly address the source of the problem? It is instructive to examine a particularly elegant solution employed by radio broadcasters.

To boost audience levels, radio shows routinely offer prizes to listeners such as concert tickets, vacations and money. Listeners tune in, wait for a signal such as a particular song and then call in hoping to win the prize. Of course, this invites an episode of severe focused loading at the switch board of the radio station as many listeners simultaneously call. The brilliantly simple way out is to announce that "caller number 9" will be the winner. This provides an incentive for listeners to randomize their call-in times—calling in too early or too late will not work—and the focused load is thereby diffused.

Of course, many of the general-purpose congestion management techniques surveyed above may also be applied to the special case of focused loading. We believe, however, that separate consideration of this special case is worthwhile, for two main reasons. First, the fact that focused loading occurs at very predictable times means that it is possible to know in advance the cases for which a specialized solution should be used. Second, the generality of the above congestion management techniques prevents them from explicitly taking into account information about agent valuation functions. Focused loading occurs because many agents prefer to use the network at the same time. This additional knowledge makes it possible to design mechanisms that collect more revenue and make fewer (*e.g.*, computational) demands on the network.

In this work we propose a game theoretic model of the problem of defocusing predictable and time-dependent focused loads. We attempt to explain why techniques such as the radio show announcement can be effective, while also contributing a formal model that permits analysis. While we do not rely on any particularly advanced results from game theory or mechanism design, we do assume that the reader is familiar with such concepts as individual rationality, risk attitudes (e.g,. risk neutrality, risk aversion) and dominant strategies. Also in the game theoretic tradition, we refer to users as agents. Good introductions to the concepts listed above are provided in [Fudenberg & Tirole, 1991; Osborne & Rubinstein, 1994].

In Section 2.2 we give a formal model of the temporal resource contention problem, define metrics for evaluating agent distributions and related notions of optimality, and specify agent utility functions. In Section 2.3 we propose a simple mechanism under which load balancing is a weak equilibrium for agents who value slots identically. We strengthen this to a strict equilibrium in Section 2.4 and also prove that this mechanism is arbitrarily close to optimal. In Sections 2.5 and 2.6 we relax the assumption that all agents have identical utility functions and present two mechanisms that balance load when only bounds on agent valuations are known. Since these mechanisms cannot take into account exactly how much each agent would be willing to pay to use the network, these mechanisms are not optimal; however, we prove a bound on their optimality which depends on the tightness of the bound on agent utility functions. If these mechanisms were used in the original case where agents value slots identically, then they too would be arbitrarily close to optimal. Finally, in Section 2.7 we summarize and compare the four mechanisms presented in this work.

2.2 Problem Definition

In order to motivate the notation that we will use throughout the chapter, it is helpful to begin with an example. Consider a network resource with a fixed number of identical time slots, where usage cost does not depend on the time slot. For example, consider a usage-based web service such as a pay-per-view streaming video service in which usage is divided into half-hour blocks from 7 PM to midnight. We assume that each agent wants to use the network during only one time slot, that each agent knows his own valuation for each slot, and that all agents' utilities are maximized by using the network during the same slot. For example, all agents might prefer to use the network from 7:00 to 7:30, having strictly monotonically-decreasing valuations for later slots as compared to earlier slots. Since time slots are priced identically, rational agents would all choose to use the network from 7:00 to 7:30, leading to a focused load. We further assume that although the capacity of the network resource is unlimited (*e.g.*, hosted on an ASP) the operator of the resource has an exogenous desire for users to de-focus their demands (*e.g.*, the ASP charges the operator for peak bandwidth used[1]).

[1]A number of proposals for usage-based pricing of bandwidth suggest charging according to the "effective" bandwidth consumed by an operator. Roughly, the effective bandwidth of a connection

2.2.1 Mechanism Characteristics

In order to spread out the focused load, the network will provide agents with an incentive to choose slots other than $\overline{s}$. In this chapter we will consider mechanisms in which agents are probabilistically spared the usage cost for the slot they choose. The cost of using the slot is waived according to a probability which depends on the slot chosen, and which is independent of the probabilities corresponding to other slots.

More formally, a mechanism Φ is defined by a tuple $\langle t, m, N, f(.)\rangle$. The network operates over t time slots, where each slot has a fixed usage cost of m, and where the set N of n agents, $a_1 \ldots a_n$, intend to use the network. Each agent a_i takes an action A_i of using a slot. The function $f : A_1 \times \cdots \times A_n \rightarrow [0,1]^n$ maps the actions taken by all agents into individual probabilities P_i that the cost of the slot chosen by a_i will be waived. Though f is specified by the mechanism, the network must draw from each P_i to determine whether the usage cost will actually be waived for each agent. Note that the probability that each slot will be made free is determined independently. By q we denote the expected number of slots that will be offered to at least one agent for free. The distribution of agents is denoted d, and so $d(s)$ is the number of agents who chose slot s.

2.2.2 Agent Characteristics

We assume that all agents are risk neutral. Agent a_i's valuation for slot s is given by an arbitrary non-negative function $v_i(s)$. Let $\overline{s}_i = \arg\max_s v_i(s)$ and $\underline{s}_i = \arg\min_s v_i(s)$. Because we are concerned with cases in which focused loading occurs we will assume that all agents have identical and unique most- and least-preferred slots, although this assumption is not required for any of our results. (If agents find several slots to be the most preferable, some amount of load balancing is likely to occur without any intervention by the network, as agents will distribute themselves across these slots.) Therefore, we define constants $\overline{s}$ and $\underline{s}$ such that for all i, $\overline{s}_i = \overline{s}$ and $\underline{s}_i = \underline{s}$. In Sections 2.3 and 2.4 we will make the assumption that all agents' valuation functions

is a value between the mean and peak bandwidths, capturing the trade-off between the long-term average amount of bandwidth used by the connection and the instantaneous peak bandwidth consumption. See, for example [Songhurst *et al.*, 1999], and the references therein.

are identical (in these sections we will use the notation v rather than v_i to describe agents' valuations). Of course this assumption is not realistic; we relax it in Sections 2.5 and 2.6. Let v^l and v^u be lower and upper bounds on all agents' valuations, respectively: *i.e.*, $\forall i, s\, v^l(s) \leq v_i(s) \leq v^u(s)$. It is important to note that these bounds apply to all agents: in our model no agent has a valuation for slot s lower than $v^l(s)$ or higher than $v^u(s)$.[2] Using this notation, the restriction on agents' valuations in sections 3 and 4 can be understood as the case where $\forall s\, v^l(s) = v^u(s)$. Finally, each agent a_i may also receive a signal from the network, denoted $\sigma(a_i)$.

In our model, the decision faced by agents is simply to choose a slot s. The space of agent strategies $\mathcal{S}$ is the space of all functions mapping from the information available to a probability distribution over slot choices. We denote an element of $\mathcal{S}$ as $S = \Pi(s)$: a distribution over slot choices. Agents are aware of the mechanism and consider it when determining their strategies. Let $\varphi \in \mathcal{S}^n$ denote a set of agent strategies, which we formally call a *strategy profile*. Let $\varphi(i)$ denote a_i's strategy under strategy profile φ, and let $\{\varphi \setminus i, S\}$ denote the strategy profile where all agents $j \neq i$ choose the strategy $\varphi(j)$ and agent a_i chooses the strategy S. We can write agent a_i's expected utility under strategy profile φ (recall that $\varphi(i)$ is a distribution over slot choices for agent a_i, and hence $\varphi(i)(s)$ is the probability that agent a_i will choose slot s under strategy profile φ):

$$u_i(\varphi) = \sum_{s_1=1}^{t} \cdots \sum_{s_i=1}^{t} \cdots \sum_{s_n=1}^{t} \Big[\big(\prod_{i=1}^{n} \varphi(i)(s_i)\big) \cdot \Big(v_i(s_i) - \big(1 - f(s_1, \ldots, s_n)_i\big) m \Big) \Big] \tag{2.1}$$

We can now give a key definition:

Definition 2.1 *φ is a* Nash equilibrium *of Φ if $\forall i, \forall S, u_i(\varphi) \geq u_i(\{\varphi \setminus i, S\})$.*

[2]While these bounds strengthen our results, the assumption that they exist is not unrealistic. The upper bound is easily justified by the fact that no agent is willing to pay an arbitrarily large amount. The lower bound is trickier, since agent a_i might simply not be interested in using some slot s (*i.e.*, $v_i(s) = 0$). However, since we're interested in defocusing the load, in practice we will be considering time slots that agents want to use. Therefore, it is not unrealistic to assume that every agent has a non-zero valuation for every slot.

Intuitively, no agent can gain by unilaterally deviating from a Nash equilibrium. This type of equilibrium is also referred to as a weak Nash equilibrium since it is possible that the agent receives equal utility from alternative strategies. When no such alternative exists, we have a *strict* Nash equilibrium:

Definition 2.2 *φ is a* strict Nash equilibrium *of Φ if $\forall i, \forall S \neq \varphi(i), u_i(\varphi) > u_i(\{\varphi \setminus i, S\})$.*

Equation (2.1) is complicated because it accounts for the calculation of the probability that slot s_i is free, starting from a strategy profile. Although this definition of utility is necessary for discussing Nash equilibria, in other parts of the chapter we will find it more convenient to take as given the same distribution p for all agents, indicating the probability of each slot being free. We can then specify an expression for a_i's expected utility for choosing slot s:

$$u_i(s) = v_i(s) - \big(1 - p(s)\big)m \tag{2.2}$$

2.2.3 Restrictions on the Class of Mechanisms

We now consider restrictions on the class of mechanisms that could be used to solve the focused loading problem, not to make the problem easier to solve, but in order to identify solutions with desirable characteristics. First, we introduce a restriction concerned with agents' incentives to participate (as discussed below, this condition is stronger than the standard mechanism design requirement of individual rationality). Next, we discuss restrictions that could arise from implementation considerations and the case of continuous pricing.

Definition 2.3 *A mechanism Φ is* participation-safe *if and only if $m \leq v^l(\overline{s})$.*

We will consider only participation-safe mechanisms in this chapter; that is, we require that the fixed usage cost for the network resource must never exceed the lower bound on any agent's valuation for his most-preferred slot. Intuitively, this means that every agent will always be able to choose at least one slot in which his payment will never exceed his valuation, and hence that it will be rational for him to participate regardless of how the mechanism assigns free slots. Observe that participation-safety implies individual rationality, because, regardless of P_i, agent a_i can choose slot $\overline{s}$ and achieve a non-negative utility. Individual rationality does depend on P_i, and thus is a weaker condition.

We do not restrict the class of mechanisms in order to simplify analysis. As it turns out, it is very easy to design and analyze mechanisms that have a fixed cost exceeding all agents' valuations, and then reward agents only when they behave as desired. Such mechanisms can have good theoretical characteristics (such as optimality, defined below) and can remain consistent with individual rationality by assuring agents non-negative expected gain. Indeed, it turns out that in what follows, everywhere we prove ε-optimality or $(c + \varepsilon)$-optimality, we could prove optimality or c-optimality, respectively, if we were not restricted to participation-safe mechanisms. However, we believe that such mechanisms would be considered unreasonable to deploy in practice despite their theoretical benefits, because they address the problem of focused loading by threatening agents with unviable alternatives—slots whose expected costs exceed agents' valuations—rather than giving agents positive incentives to behave as desired.

Because of the difficulty of implementing complex protocols on a highly-loaded network resource, it is worthwhile to consider various other restrictions on the class of mechanisms. For example, it may or may not be possible to reimburse agents after all agents have chosen a slot, as opposed to doing so after each agent chooses. Also, it may or may not be permissible for f to depend on what slots agents chose, as this would require that information be stored for each agent, and again that billing be deferred until after all agents have selected slots. In some settings it might not be reasonable for the network to give signals to agents; in other cases, it would be possible to give signals but not to record which signals were given to which agents. The significance of the time, space and communication complexity of the mechanism may also vary depending on the setting. We discuss these and other trade-offs in Section 2.7.

Also, it might appear that more powerful mechanisms could be designed if prices could be varied arbitrarily, as opposed to our model in which slots must be priced at either m or 0. In fact, since we assume that agents are risk-neutral, agents will be indifferent between any slot priced on the range $[0, m]$ and the same slot made free with an appropriate probability. Furthermore, m can be increased arbitrarily. In the case of risk-averse agents, such 'continuous pricing' *would* be useful: our results throughout this chapter hold for risk-averse agents if and only if this sort of continuous pricing scheme is used. We have chosen not to emphasize continuous pricing because it would be likely to make greater computational and communication demands on the network; however, all our results are compatible with such a scheme, and furthermore our bounds on q and m (see, e.g, equations (2.7), (2.8), and (2.9)) may be dropped in this case.

2.2.4 Evaluating Outcomes

The network has two aims: to balance the load caused by the agents' selection of slots and to collect as much revenue as possible. We denote the network's expected revenue given a mechanism Φ and equilibrium φ as $E[R|\Phi, \varphi]$. The network collects a payment of m from each participating agent except for those who receive free slots. Expected revenue is given by:

$$E[R|\Phi,\varphi] \quad = \quad \sum_{i=1}^{n}\sum_{s_1=1}^{t}\cdots\sum_{s_n=1}^{t}\Big[\big(\prod_{i=1}^{n}\varphi(i)(s_i)\big)\cdot\big(1 \quad - \quad f(s_1,\ldots,s_n)_i\big)m\Big] \tag{2.3}$$

We define g as the monetary value to the network of the variance of load across the set of time slots. Lower variance corresponds to a more even load and thus to a higher dollar value; thus g must decrease strictly as variance increases. We will say that load is balanced when g is maximized, which corresponds to minimal variance. We define the *superlinear summation* class of functions to be the set of functions in which $g(d) = -\kappa \sum_i h(d(i))$, where h is superlinear in $d(i)$ and κ is a constant that is used to indicate the relative importance of load balancing to the network. Note that this measure is only reasonable if we assume that each agent consumes about the same amount of load. The expected value of load balancing is given by:

$$E[g|\varphi] = \sum_d g(d) Prob(d|\varphi) \tag{2.4}$$

Maximizing revenue and maximizing g are conflicting goals, as it costs the network more to induce an agent to choose slot $\underline{s}$ than to choose slot $\overline{s}$. Indeed, note that revenue is maximized in the original focused loading equilibrium when all agents choose $\overline{s}$ and $\forall i\, P_i = 0$. According to our problem definition, agents are *willing* to distribute themselves this way, and thus this equilibrium can be achieved without waiving any agents' usage fees. In some systems this could be a desirable outcome; however, we have assumed that the mechanism designer would prefer at least some balancing of the load. The network must therefore trade off quality of load balancing against expected revenue; the degree of trade-off desired may be specified through the choice of κ. Given definitions of the expected values R and g, we can define z, the network operator's evaluation of equilibrium φ of mechanism Φ:

$$z(\Phi, \varphi) = E[R|\Phi, \varphi] + E[g|\varphi] \tag{2.5}$$

It will be useful to define the best possible distribution of agents given a free slot distribution that applies to all agents. Imagine a mechanism Φ_{all} in which all strategy profiles are in equilibrium, and $P_i = p(A_i)$ (*i.e.*, the probability that a slot will be free for agent a_i depends only on his action). Intuitively, this is the best distribution of agents for the mechanism, given the constraint that the free slot distribution must be the same for all agents.

Definition 2.4 *A distribution d is* ideal *for $p(s)$ if and only if an equilibrium φ which deterministically results in distribution d maximizes $z(\Phi_{all}, \varphi)$.*

Note that this expression may not have a unique maximum. We will denote an ideal distribution d as d^*.

Next, we define the optimality of an equilibrium under a mechanism. Essentially, an equilibrium of a given mechanism is optimal if there does not exist another equilibrium of any other mechanism that yields a higher expected value of z.

Definition 2.5 *A mechanism-equilibrium pair* (Φ, φ) *is optimal if and only if for all other pairs* (Φ', φ')*,* $z(\Phi, \varphi) \geq z(\Phi', \varphi')$*, where* n *is held constant.*

This definition of optimality is problematic when agents have different valuation functions that are not known by the network—the case we take up in Sections 2.5 and 2.6. An optimal mechanism for this case would have to set each agent's expected payment to exactly his valuation for any slot chosen, by constructing a different P_i for each agent. For every set of agents there does exist a set of such mechanisms. However, it is impossible to select such a mechanism based on the information available; furthermore such a mechanism will violate our restriction that it be participation-safe, because an agent a_i who chooses slot $\overline{s}$ is charged $v_i(\overline{s})$, which can be exceed $v^l(\overline{s})$. To overcome this difficulty we provide an alternate notion of optimality that bounds the average loss per agent as compared to an optimal mechanism:

Definition 2.6 *A mechanism-equilibrium pair* (Φ, φ) *is c-optimal if and only if for all other pairs* (Φ', φ')*,* $z(\Phi, \varphi) + cn \geq z(\Phi', \varphi')$*, where* n *is held constant and* $c > 0$*.*

For convenience, we will also make use of the term *[c-]optimal* to refer to equilibria alone, in cases where the mechanism giving rise to the equilibrium is unambiguous.

Definition 2.7 *An equilibrium* φ *is [c-]optimal if* φ *is an equilibrium of mechanism* Φ*, and* (Φ, φ) *is [c-]optimal.*

We call φ' where all agents choose the same slot a *focused-loading equilibrium*. We assume that g and v do not take values that would cause φ' to be optimal. This assumption is only required for our proof of Theorem 2.9, but it is a reasonable one for us to make since if φ' *were* optimal, we would have no problem to solve in the first place.

2.3 Preselection Mechanism

In this section we consider a simple mechanism, designed to make agents indifferent between all time slots despite their initial preferences. This mechanism will be formally referred to as Φ_1, and informally called 'preselection', since it decides which slots will be free before observing the actions of the agents. This mechanism is unrealistic in several ways, and we do not discuss it here in order to propose that it should be used in practice. Indeed, such a mechanism is an obvious first approach to the problem of focused loading, and so it is important to demonstrate its insufficiency. Furthermore, the exposition of this mechanism will prove useful as a starting point for the discussion of more sophisticated mechanisms.

$\Phi_1$works as follows:

1. The network determines free slots by drawing from p. (Thus, $P_i = p(A_i)$.)
2. Agents choose a slot.

2.3.1 Equilibria

We know from the definition of the problem that when there is no chance that they will win a free slot agents prefer slot $\overline{s}$ to slot $\underline{s}$. We can overcome this preference by selecting an appropriate $p(s)$. An agent's expected utility is given by $u_i(s) = v(s) - (1 - p(s))m$. Recall that we assume $v^l = v^u$ until Section 2.5; here we use (unsubscripted) v to denote the valuation function that all agents share. We can make agents indifferent between slots by requiring that all time slots will have the same expected utility for agents: that is, that the expected utility derived from each time slot is equal to the average expected utility over all time slots. This is expressed by the equation $v(s) - \big(1 - p(s)\big)m = \frac{1}{t}\sum_i \Big(v(i) - \big(1 - p(i)\big)m\Big)$. Algebraic manipulation and $q = \Sigma_s p(s)$ give us:

$$p^*(s) = \frac{\frac{1}{t}\big(qm + \sum_i v(i)\big) - v(s)}{m} \tag{2.6}$$

Observe that since free slots are free for all agents, q represents the expected number of free slots. Because we will find this probability distribution useful throughout the chapter, we have given it a name: p^*.

If free slots are awarded according to p^*, it is a weak Nash equilibrium for all agents to select a slot uniformly at random. We will call this equilibrium φ_1. Consider the case where all other agents play according to φ_1, and one remaining agent a_i must decide his strategy. Since the choice of any slot entails equal utility on expectation, a_i can do no better than to randomly pick a slot. Again, φ_1 is only a weak equilibrium: indeed, there is no strategy a_i could follow that would make him worse off.

We now make several remarks about the preselection mechanism. First, note that the above analysis assumes that a_i is risk-neutral. If a_i is risk-averse, he will prefer slot $\overline{s}$, since it gives the largest fixed payment, $v(\overline{s})$. Second, this mechanism is not susceptible to collusion, because each agent is indifferent between all pure strategies regardless of the actions of other agents. Finally, since *all* strategy profiles are weak equilibria under the preselection mechanism, it would be reasonable to ask why we pay special attention to φ_1. It may be argued that randomization is a "natural" response to indifference, and so we will consider this as a primary case in the next subsection; however, none of our results depend on the assumption that agents will choose this strategy.

2.3.2 Bounds on q and m

It appears that deviation from φ_1 will never be profitable for agents, since we have guaranteed that all slots provide the same expected utility. Consider the most profitable deviation, from $\underline{s}$ to $\overline{s}$. We have claimed that the utility of both slots is the same: $v(\overline{s}) - \big(1 - p(\overline{s})\big)m = v(\underline{s}) - \big(1 - p(\underline{s})\big)m$. However if qm is too small or too large, $p(\underline{s}) - p(\overline{s}) > 1$ will hold. Since we want to interpret $p(\underline{s})$ and $p(\overline{s})$ as probability measures, we must add the constraints $p(\overline{s}) \geq 0$ and $p(\underline{s}) \leq 1$. Without these constraints, the equation for p^* still makes sense if we consider continuous pricing rather than our default model of free/non-free slots; $p > 1$ corresponds to an expected slot cost of less than zero (paying agents to choose a slot) while $p < 0$ corresponds to an expected slot cost of more than m. Substituting $p(\overline{s}) \geq 0$ into Equation (2.6) and rearranging, we get:

$$q \geq \frac{tv(\overline{s}) - \Sigma_i v(i)}{m} \tag{2.7}$$

For the second condition, we require that $p(\underline{s}) \leq 1$, which gives us:

$$q \leq \frac{t\big(v(\underline{s}) + m\big) - \Sigma_i v(i)}{m} \tag{2.8}$$

We must also ensure that a value of q exists for a given m and v. Intersecting the two bounds and simplifying, we get:

$$m \geq v(\overline{s}) - v(\underline{s}) \tag{2.9}$$

Indeed, if $m < v(\overline{s}) - v(\underline{s})$ then if an agent were certain to win a free slot in $\underline{s}$ and guaranteed never to win a free slot in $\overline{s}$, he would still prefer $\overline{s}$ to $\underline{s}$.

2.3.3 Maximizing Revenue

Equation (2.3) gave a general expression for $E[R|\Phi, d]$. However, under equilibrium φ_1 all agents randomly select a slot, which allows us to write an expression for $E[R|\Phi_1, \varphi_1]$ that does not include a summation. In φ_1 expected revenue is given by the percentage of non-free slots times cost per slot times number of agents:

$$E[R|\Phi_1, \varphi_1] = \left(1 - \frac{q}{t}\right) mn \tag{2.10}$$

Increasing m will increase expected revenue; however, recall that we require that the mechanism be participation-safe, and hence that $m \leq v^l(\overline{s})$. Regardless of the particular value of m, reducing q (the expected number of free slots) will increase expected revenue.

We will now show how the network can maximize revenue. We define v_{avg} as $\frac{1}{t}\sum_s v(s)$. The requirement that an agent's utility for slot s must be greater than or equal to zero—*i.e.*, that $v(s) - \big(1 - p(s)\big)m \geq 0$—can be rewritten, substituting in p^*, as $v_{avg} - (1 - \frac{q}{t})m \geq 0$. The seller's revenue will be maximized when all agents get zero utility. Thus we must have:

$$\left(1 - \frac{q}{t}\right) m = v_{avg} \tag{2.11}$$

We substitute in the lower bound for q from Equation (2.7): *i.e.*, $q = \frac{1}{m}\big(tv(\overline{s}) - \Sigma_i v(i)\big)$. Rearranging for m, we get $m = v(\overline{s})$. This satisfies Equation (2.9) and ensures that the mechanism is participation-safe, so we are done.

This is intuitive because when we minimize q we set $p(\overline{s}) = 0$. We know that agents are indifferent between all slots, and so agents will be willing to choose any slot when the cost of $\overline{s}$ does not exceed their valuation. We thus set $m = v(\overline{s})$ and (plugging m into the lower bound on q) $q = t\big(1 - v_{avg}/v(\overline{s})\big)$.

We have shown that each agent can be made to pay an expected amount exactly equal to his utility for any slot he chooses. However, φ_1 is not guaranteed to achieve an ideal distribution of agents, and therefore φ_1 is not optimal. The easiest way to show this is to present another equilibrium of the preselection mechanism that *is* optimal.

2.3.4 Optimal Equilibria

Consider an equilibrium in which each of the agents deterministically chooses one slot. (Recall that *any* strategy is rational under Φ_1, and thus that any set of strategies is a weak equilibrium.) In one such equilibrium, agents deterministically choose slots so that the distribution of all agents is ideal; we will call this equilibrium φ_1^*. Unsurprisingly, we can show:

Theorem 2.8 (Φ_1, φ_1^*) *is optimal.*

Remark. Recall that a mechanism-equilibrium pair is optimal when there does not exist another mechanism that has an equilibrium giving rise to a distribution that yields a higher value according to the evaluation function z.

Proof. Let $E[R_i|\Phi, \varphi]$ be the expected revenue extracted from agent a_i, given mechanism Φ and equilibrium φ. First, we prove by contradiction that Φ_1 yields at least as large a z as any other Φ, both given the *same* equilibrium of the respective mechanisms. Assume that there exists a pair (Φ, φ) such that $z(\Phi, \varphi) > z(\Phi_1, \varphi)$. Since the equilibrium is constant, we can expand z on both sides and simplify to get $E[R|\Phi, \varphi] > E[R|\Phi_1, \varphi]$, which implies $E[R_i|\Phi, \varphi] > E[R_i|\Phi_1, \varphi]$ for at least one agent a_i. Φ_1 sets values of p, q and m so that for all slots agent a_i's expected utility is $v(s) - E[R_i|\Phi_1, \varphi] = 0$. Thus for Φ we have $\forall s\, u_i(s) < 0$, implying that Φ is non-participation-safe, a contradiction.

Second, we consider the case where Φ and Φ_1 give rise to different equilibria. As described above, under φ_1^* agents deterministically distribute themselves so as to give rise to the distribution d^*. Recall that d^* is an ideal distribution: $\forall \varphi \left(z(\Phi_1, \varphi_1^*) \geq z(\Phi_1, \varphi) \right)$. Thus, $\forall \Phi, \varphi \; z(\Phi_1, \varphi_1^*) \geq z(\Phi_1, \varphi) \geq z(\Phi, \varphi)$. ■

The equilibrium φ_1^* is optimal, but it is extremely unlikely that it would arise through the choices of real agents. As mentioned above, the fact that agents are indifferent between all slots means that *every* combination of agent strategies is a weak equilibrium. In fact, the preselection mechanism gives rise to many equilibria that minimize $g(d)$. For example, the case in which all agents choose slot $\overline{s}$ is a weak equilibrium. Since discouraging focused loading is the purpose of the preselection mechanism, it is undesirable to find that such behavior remains an equilibrium! However, this drawback is inherent to the setting as we have modeled it so far; a preselection mechanism can only yield weak equilibria or focused-loading equilibria.

Theorem 2.9 *When agents have identical utility functions and no signals are given to agents and the network preselects p before agents move, all equilibria are either weak or focused-loading.*

Remark. Intuitively, this proof shows that under the conditions of the preselection mechanism any incentive given to one agent is given to all the agents, and that the mechanism designer must therefore choose between encouraging all agents to choose the same slot and making all agents indifferent between a set of slots.

Proof. Consider two agents a_i and a_j, without restriction. The network has only three choices with respect to a_i's preferences:

1. a_i strictly prefers some slot s_k to every other slot. However, every other agent a_j has the same preference. Therefore, no agents will choose any other slot. This is a strict equilibrium, but it is also a focused-loading equilibrium.

2. a_i will (non-strictly) prefer some slot s_k to all other slots: he will strictly prefer s_k to s_l, and will be indifferent between s_k and at least one other slot. Thus no agents will choose slot s_l, and g will not be minimized. Any set of mixed

strategies over slots between which agents are indifferent will constitute a weak equilibrium.

3. a_i is indifferent between all pairs of slots s_k and s_l. In this case a_i receives the same payment regardless of his action, so randomizing uniformly over all the slots is not a dominated strategy. Indeed, randomization is a weak, load-balancing equilibrium, as shown above.

The only strict equilibrium is a focused-loading equilibrium; all other equilibria are weak. ■

In fact, we can show another negative result: there does not exist an optimal mechanism that is participation-safe and that gives rise to a strict equilibrium.

Theorem 2.10 *There does not exist an optimal* (Φ, φ) *for which* φ *is a strict equilibrium and* $m \leq v(\overline{s})$.

Proof. We will prove this statement by contradiction. Assume that there exists an optimal (Φ, φ) in which φ is a strict equilibrium. Since φ is a strict equilibrium, the difference (call it x) between expected utility from slot s and the highest expected utility of any other slot must be positive. By the assumption that $m \leq v(\overline{s})$, deviation to $\overline{s}$ would result in no less than 0 utility. Thus by strictness of φ, agents in slots $s \neq \overline{s}$ have positive expected utility of x. If we create Φ' by altering P_i so that the expected utility of s is decreased by x, then the revenue is increased, and it is still an equilibrium (albeit weak) for a_i to select slot s. The fact that revenue is higher in (Φ', φ) than (Φ, φ) but that both give rise to the same distribution contradicts the claim that (Φ, φ) is optimal. ■

Theorem 2.10 shows that strict, optimal equilibria do not exist for participation-safe mechanisms. However, if we allow networks with different characteristics than those we allowed in this section, we can see that it is possible to get close to a strict, optimal equilibrium when agents have identical utility functions and no signals are given to agents, and p depends on the agents' actions. Intuitively, consider a mechanism that sets $p = (1 + \varepsilon)p^*$ if agents achieve an ideal distribution, and $p = 0$

otherwise. Further, consider a set of (pure) agent strategies where agents *happen* to distribute themselves according to d^* for $p^*(s)$. This is an equilibrium because agents are penalized for deviating. Intuitively, it is nearly optimal because agents achieve an ideal distribution with respect to the mechanism, and the probability of awarding free slots is arbitrarily close to the probability from the optimal mechanism-equilibrium pair described in theorem 2.8. However, it would be extremely difficult for agents to coordinate to this equilibrium in real play. In the next section we will show how the use of a non-binding coordination phase before the selection of slots can help agents to reach strict, nearly-optimal equilibria.

2.4 Bulletin Board System Mechanism

In this section we assume that agents are given a *bulletin board system*: a forum in which all communications are visible to all agents and the identity of agents is associated with their transmissions. For simplicity, we allow a very limited form of communication: agents indicate the slot that they intend to choose. We assume that agents do not all indicate slots at the same time; rather, they indicate sequentially during the first phase. Let $d_j(s)$ denote the number of agents who have indicated that they will choose slot s after a total of j agents have posted to the bulletin board. d^* will again be the ideal distribution for $p*(s)$. Agents' communications through the bulletin board are *cheap talk*: a technical term that indicates that these communications are not binding in any way. Even so, the bulletin board can help agents to coordinate on desirable equilibria. Mechanism Φ_2 follows:

1. The network picks "potentially free"[3] slots according to $(1+\varepsilon)p^*$.

2. Agents communicate through the bulletin board.

3. Agents choose time slots.

4. If $d = d^*$, then "potentially free" slots are made to be free. That is, $P_i = p^*(A_i)$. Otherwise, all agents are made to pay for their slots ($P_i = 0$).

[3]We redefine q as the expected number of "potentially free" slots; the same redefinition is required for Section 2.6.

2.4.1 Equilibria

A strict equilibrium in Φ_2, which we call φ_2, is for the i^{th} agent to indicate on the bulletin board a slot s such that $d_{i-1}(s) < d_i^*(s)$, and ultimately to choose that slot s. Consider the case where all other agents follow φ_2 and agent a_i must decide his strategy. If a_i cooperates and chooses slot s then the distribution of agents will be d^* and so a_i will receive an expected utility of $v(s) - \big(1 - (1+\varepsilon)p^*(s)\big)m$. If a_i defects to slot s', one of two cases will result. In the first case, agents indicating their choices after a_i will compensate for his deviation by choosing different slots; thus a_i will receive the same expected utility as he would have received if he had not deviated. In the second case, a_i will be late enough in the sequence of agents indicating their choices that the agents who indicate after him will be too few to bring the distribution back to d^*. In this case a_i will receive an expected utility of $v(s') - m$. The key point is that a_i does not know the total number of agents, and so he must assign non-zero probability to the second case, regardless of the number of agents who have already indicated. Furthermore, we must show that a_i will choose the slot he indicated on the bulletin board even though his selection was not binding. If all other agents follow φ_2 then there is clearly no incentive for a_i to choose a different slot than he indicated, because that would certainly prevent $d = d^*$ and reduce his payoff. Therefore φ_2 is strict as long as $v(s) + (1+\varepsilon)p^*(s)m > v(s')$ for all s, s' such that $1 \le s, s' \le t$. Simplifying, we derive the conditions similar to those described in Section 2.3.

$$\frac{tv(\overline{s}) - \Sigma_i v(i)}{m} \le q \le \frac{t\big(v(\underline{s}) + \frac{m}{1+\varepsilon}\big) - \Sigma_i v(i)}{m} \tag{2.12}$$

Again, we must intersect the two bounds to get a bound on m, which we combine with the constraint on participation-safe mechanisms:

$$(1+\varepsilon)\big(v(\overline{s}) - v(\underline{s})\big) \le m \le v(\overline{s}) \tag{2.13}$$

This equilibrium relies on the fact that each agent can choose a slot as if he were the last agent and achieve the distribution d^*, even if all agents before him chose slots in this same way. We prove that this greedy approach works in Section 2.4.2.

An analysis of the possibility of collusion in the bulletin board mechanism is not appropriate, because agents are already encouraged to coordinate with each other. Any agent or cartel of agents who deviated would hurt themselves along with all other agents.

It is well known that any game having an equilibrium arising from cheap talk coordination has other equilibria in which agents ignore the cheap talk [Crawford & Sobel, 1982]. The bulletin board mechanism is no exception. All agents choosing $\overline{s}$ (focused loading) is an equilibrium when the resulting d could not be transformed into d^* by one agent choosing a different slot. Note, however, that φ_2 Pareto-dominates all equilibria where the cheap talk is ignored and a different distribution results.

2.4.2 Greedy Assignment of Slots

In φ_2 each agent chooses a slot that would result in an optimal distribution if he were the last agent to post to the bulletin board. For this reason it is important to show that we can assign slots to agents greedily, with the guarantee of achieving the ideal distribution for whatever number of agents eventually participate.

We must introduce new notation to describe changes as each agent chooses a slot in turn. (Readers who do not intend to read the proof for lemma 2.11 can safely skip to section 2.4.3.) First, we will subscript d to indicate the total number of agents in the distribution, so that we can describe the distributions that result after only a subset of agents have chosen slots. By d_i^* we denote the optimal distribution of i agents. Second, we define $\Delta(d_i, s)$ to be the increase in z if one agent is added to slot s, relative to d_i. Define the decomposition $\Delta(d_i, s) = \Delta_E(d_i, s) + \Delta_g(d_i, s)$, where $\Delta_E(d_i, s)$ is the increase in $E[R|\Phi, d_i]$, and $\Delta_g(d_i, s)$ is the increase in $g(d_i)$. In equilibrium $\Delta_E(d_i, s)$ does not depend on d_i, but only on $p(s)$ and m. (We assume here that p does not depend on d_i^*.) Two properties follow from the fact that g is *superlinear summation*:

1. $\Delta_g(d_i, s)$ is strictly monotonically decreasing in $d_i(s)$

2. $\Delta_g(d_i, s) = \Delta_g(d'_j, s)$ for all distributions d'_j where $d'_j(s) = d_i(s)$

Since Δ_E does not depend on d_i, Δ also has these properties.

We now describe a function γ: let $\gamma(i)$ represent the slot number that will be assigned to a_i, where a_i is the i^{th} agent to register. Let $d_i^\gamma(s)$ be the number of times s occurs in $\{\gamma(1), \ldots, \gamma(i)\}$. We note that $\forall s \Delta(d_0^\gamma(s)) = 0$. We can now inductively define γ: $\gamma(i) = \arg\max s\Delta(d_{i-1}^\gamma(s))$.

Lemma 2.11 *$\forall i$ d_i^γ is ideal under Φ_2.*

Remark. This lemma demonstrates that greedy assignment of slots to agents leads to an ideal distribution when we assign slots according to γ as defined above. Recall that an ideal distribution is defined in Definition 2.4.

Proof. Define $d_i \geq d'_j$ as $\forall s\ d_i(s) \geq d'_j(s)$. We will prove the following statement that is stronger than the theorem: $\forall j, i \geq j$, there exists an ideal distribution d_i^* such that $d_i^* \geq d_j^\gamma$.

We will first prove this statement by induction on j. The base case, where $j = 0$, trivially holds because $\forall s\ d_0^\gamma(s) = 0$.

For the inductive step, assume that there exists a d_i^* for all $i \geq j$ such that $d_i^* \geq d_j^\gamma$, in order to prove that there exists a d_i^* for all $i \geq j+1$ such that $d_i^* \geq d_{j+1}^\gamma$. From the inductive assumption we know that there exists a $d_i^* \geq d_j^\gamma$ for each $i \geq j+1$. Let $s_k = \gamma(j+1)$: hence $s_k = \arg\max s\Delta(d_j^\gamma, s)$.

We now prove that there exists an ideal distribution $d_i'^*$ consistent with this greedy choice. If $d_i^*(s_k) \geq d_j^\gamma(s_k) + 1$, then $d_i'^* = d_i^*$. Otherwise, $d_i^*(s_k) = d_j^\gamma(s_k)$. Consider a slot s_l where $d_i^*(s_l) \geq d_j^\gamma(s_l) + 1$. Let $\Upsilon(d, s, c)$ be distribution d but with c agents added to slot s. Let $d' = \Upsilon(d_i^*, s_l, -1)$, and let $d'' = \Upsilon(d', s_k, 1)$. We know from the first property of Δ that $\forall s\ (\Delta(d', s) \leq \Delta(d_j^\gamma, s))$, since $d' \geq d_j^\gamma$. Similarly, from the second property of Δ we know that $\Delta(d', s_k) = \Delta(d_j^\gamma, s_k)$, since $d'(s_k) = d_j^\gamma(s_k)$. Therefore, $s = s_k$ maximizes $\Delta(d', s)$. This implies that $z(\Phi, d'') \geq z(\Phi, \Upsilon(d', s_l, 1))$. Since $\Upsilon(d', s_l, 1) = d_i^*$ is ideal, d'' must also be ideal. Since $d'' \geq d_{j+1}^\gamma$, we have proven the inductive step. ∎

2.4.3 ε-Optimality

Although theorem 2.10 showed that the bulletin board mechanism cannot be optimal, it turns out that it can be made arbitrarily close to optimal. We now show that there exists no other equilibrium of any other mechanism which will yield a value of z larger than $z(\Phi_2, \varphi_2) + \varepsilon$ for arbitrarily small ε.

Theorem 2.12 *(Φ_2, φ_2) is ε-optimal.*

Remark. This is a key result, because it shows that we can get arbitrarily close to an optimal equilibrium with a mechanism that could actually be used in practice. Furthermore, the fact that the equilibrium is strict is encouraging, because it means that an agent could not reduce z by deviating from φ_2 without also reducing his own utility.

Proof. First, we prove by contradiction that Φ_2 yields z that is within $n\varepsilon$ of any other Φ, both given the *same* equilibrium. Assume that there exists a pair (Φ, φ) such that $z(\Phi, \varphi) > z(\Phi_2, \varphi) + n\varepsilon$. Since the equilibrium is the same for both mechanisms, we can expand z on both sides and simplify to get $E[R|\Phi, \varphi] > E[R|\Phi_2, \varphi] + n\varepsilon$, which implies $E[R_i|\Phi, \varphi] > E[R_i|\Phi_2, \varphi] + \varepsilon$ for at least one agent a_i. Φ_2 sets values of p, q and m so that for all slots agent a_i's expected utility is $v(s) - E[R_i|\Phi_2, \varphi] = \varepsilon$. Thus for Φ we have $\forall s\ u_i(s) < 0$, implying that Φ is non-participation-safe, a contradiction.

Second, we now consider the case where Φ and Φ_2 have different equilibria. As shown above in lemma 2.11, the ideal distribution d^* is achieved by (Φ_2, φ_2), hence $\forall \Phi, \varphi\ z(\Phi_2, \varphi_2) \geq z(\Phi_2, \varphi) \geq z(\Phi, \varphi) - n\varepsilon$. ■

2.4.4 Implementation Considerations

We point out that ε-optimality means that the mechanism can lose ε per agent; in practice, ε would have to be large enough to overcome agents' indifference between nearly-identical payoffs and encourage them to coordinate.

Although we speak about agent strategies throughout this chapter, it is worthwhile to note that in a real system these strategies would probably be implemented in software that most users would not be able to change easily. Of course, this is not an argument against equilibrium analysis or the careful design of economic mechanisms. If agents could gain by deviating, there would be an incentive for users to change their software, and once software has been modified it is easily redistributed. However, the fact that the mechanism designer could in many cases distribute client software is significant because it can act as a coordination device: agents' common knowledge of using the same software could help them to coordinate to an equilibrium the mechanism designer has preselected. Although the bulletin board mechanism gives rise to non-ε-optimal equilibria, these might be avoided if client software helped agents to coordinate to φ_2.

2.5 Collective Reward Mechanism

We now consider the more general and realistic case where each agent may have a different v_i, bounded by v^l and v^u, as described in Section 2.2. Recall that since the network does not know each agent's v, we can no longer tune m, q, and p to extract the maximum amount of revenue from each agent.

In this section we also allow the network to give signals to agents, to allow the agents to coordinate to a desirable equilibrium; we also show how collective reward may be used to prevent agents from deviating. We define mechanism Φ_3 as follows:

1. Each agent indicates that he will participate.
2. The network gives a signal to each agent from $\{1, \ldots, t\}$.
3. Agents choose time slots.
4. The network determines whether each slot will retroactively be made free.

In this mechanism, the chance that slot s will be free, $p(s)$, depends on the number of agents who chose slot s, $d(s)$. Let $count(s)$ be the number of agents who were given the signal s. Define $d^+(s) = d(s) - count(s)$. For the collective reward mechanism Φ_3:

$$p(s) = \begin{cases} p^b(s) & \text{if } d^+(s) \le 0 \\ 0 & \text{if } d^+(s) > 0 \end{cases} \tag{2.14}$$

Thus $P_i = p^b(A_i)$ if $d^+(s) \le 0$ and $P_i = 0$ otherwise, where $p^b(\cdot)$ is defined below.

We will assign signals to agents so that $count(s) = d^*(s)$, where d^* is now ideal for $p^b(s)$. The idea of this mechanism is that agents who choose the slot s to which they are assigned will get that slot free with probability $p^b(s)$, and agents who deviate to another slot will pay m. The $p(s)$ used for this mechanism will thus differ from $p(s)$ for the previous two mechanisms. The intuitive reason for the change is that in Φ_1 and Φ_2 we used p to make agents indifferent between all slots. Now, however, we use p so that agents will not deviate from an assignment to a particular slot. We will construct p^b so that each agent a_i will choose his assigned slot even when a_i has the lowest possible valuation for the slot corresponding to his signal, and the highest possible valuation for $\overline{s}$, the most profitable slot to which he could deviate. When an agent is assigned a slot $s \neq \overline{s}$, this condition can be formalized as:

$$v^l(s) - \big(1 - p^b(s)\big)m = v^u(\overline{s}) - m + \varepsilon \tag{2.15}$$

Here as before ε is a small, positive value used to make agents strictly prefer the slot to which they are assigned. It can be interpreted as an offset to v^u, giving us a strict upper bound on agents' utilities. If we make an agent with this impossibly high valuation for slot $\overline{s}$ indifferent between his assigned slot and $\overline{s}$, then any agent who actually plays the game must prefer his assigned slot. We can now derive p^b:

$$p^b(s) = \begin{cases} \frac{v^u(\overline{s}) - v^l(s) + \varepsilon}{m} & \text{if } s \neq \overline{s} \\ 0 & \text{if } s = \overline{s} \end{cases} \tag{2.16}$$

The case of $s = \overline{s}$ is considered separately because an agent assigned to this slot has no incentive to deviate. Note that if $v_i(s) = v_i(\overline{s})$ is possible for an $s \neq \overline{s}$, then we would have to change the definition of p^b to maintain a strict equilibrium, giving ε' probability of awarding $\overline{s}$ free.

We now need to give bounds on m. The condition that $p^b(s) \leq 1$ can be rewritten, combined with the requirement that the mechanism be participation-safe, as:

$$v^u(\overline{s}) - v^l(\underline{s}) + \varepsilon \leq m \leq v^l(\overline{s}) \tag{2.17}$$

For Φ_3 q is defined as:

$$q = \sum_{i \neq \overline{s}} \Big(\frac{v^u(\overline{s}) - v^l(i) + \varepsilon}{m} \Big) \tag{2.18}$$

To maximize expected revenue, the collective reward mechanism sets m to its upper bound of $v^l(\overline{s})$.

2.5.1 Equilibria

An equilibrium φ_3 is for each agent a_j to select the slot corresponding to his signal.[4] Consider the case where all other agents follow this strategy, and one remaining agent a_i decides his strategy. If agent a_i selects slot s as above, then his expected utility is $u_i(s) = v_i(s) - \big(1 - p^b(s)\big)m$. Deviating to even the best slot only gives him $u_i(\overline{s}) = v_i(\overline{s}) - m$. We have defined p^b so that in this case a_i strictly prefers slot s.

There are no equilibria of the collective reward mechanism for which $d \neq d^*$. Consider any distribution of agents such that $d \neq d^*$. There must be some s_1 such that $d^+(s_1) < 0$, and some other s_2 such that $d^+(s_2) > 0$. An agent in s_2 thus has no chance of a free slot, and he receives utility of at most $v_i(\overline{s}) - m$. If he switches to s_1, then his probability of receiving a free slot becomes $p^b(s_1)$ because $d^+(s_1) \leq 0$. Since p^b is constructed so that this agent receives more utility, on expectation, than $v_i(\overline{s}) - m$, he has incentive to move to slot s_1. However, there do exist equilibria in which agents do not select slots corresponding to the signals they receive. For example, consider the case where agent a_i deterministically selects the slot $\sigma(n+1-i)$. (Note that this could occur even if agent a_i did not *know* what signal agent a_{n+1-i} receives.) In this case the distribution of agents is d^*, and so the analysis above demonstrates that all agents have a disincentive to deviate. Another example is where all agents select the slot corresponding to their signals except where agent a_i chooses slot $\sigma(j)$ and agent a_j chooses slot $\sigma(i)$.

[4]This note is intended for readers familiar with game theory. Consider the space of all functions $H : \mathbb{N} \rightarrow \{1, \ldots, t\}$ mapping from agent names to suggested slots. Let *Prob* be a probability distribution over all functions $h \in H$ that give rise to the agent distribution d^*. If signals are assigned based on an h drawn from *Prob* then φ_3 can easily be formulated as a correlated equilibrium. However, for ease of exposition and to emphasize the sequential assignment of agent signals for implementation reasons, we do not make further use of this formulation.

Harmful collusion is not possible under the collective reward mechanism. A single agent who deviates from φ_3 can harm other agents by denying them a chance at a free slot. However, no set of agents is able to *improve* other agents' chance of getting a free slot, and so there is no way that a cartel of agents could benefit from colluding.

Theorem 2.13 (Φ_3, φ_3) *is c-optimal for* $c = \max_s\big(v^u(s) - v^l(s)\big)+\varepsilon$.

Remark. Because it depends on bounds rather than on agents' actual valuations, φ_3 is not optimal. However, this theorem shows that we can prove a bound on the optimality of φ_3, showing that the network can lose no more than $\max_s\big(v^u(s) - v^l(s)\big)+\varepsilon$ in revenue from each agent.

Theorem 2.14 (Φ_3, φ_3) *is c-optimal for* $c = \max_s\big(v^u(s) - v^l(s)\big)+\varepsilon$.

Proof. Define $v^{l+c-\varepsilon}(s) = v^l(s)+c-\varepsilon$: an upper bound on v^u and thus on all possible v functions for agents. We now define variants of Φ_3 based on different agent v functions: Φ_3^a when agents have different, arbitrary v functions, and Φ_3^l and $\Phi_3^{l+c-\varepsilon}$ for the cases when all agents' functions are v^l and $v^{l+c-\varepsilon}$, respectively. In each variant we assume that the network has full knowledge of agents' valuations and can set different p's for each agent. Let d^a, d^l, and $d^{l+c-\varepsilon}$ be the corresponding ideal distributions arising from φ_3 in their respective mechanisms. The revenue extracted from each agent in equilibrium of Φ_3^a , Φ_3^l or $\Phi_3^{l+c-\varepsilon}$ is: $(1-p(s))m = (1 - \frac{v(\overline{s})-v(s)+\varepsilon}{v(\overline{s})})v(\overline{s}) = v(s) - \varepsilon$. We also make the change that each of these variants of Φ_3 sets $\varepsilon = 0$ when it determines p^b. This has the consequence that equilibrium φ_3 still holds but is no longer strict. Each variant will then extract the full $v(s)$ from each agent in φ_3. Each of these mechanism-equilibrium pairs is optimal, following an argument analogous to the one given in the proof of theorem 2.8 (not given here): the mechanism makes each agent pay exactly his valuation, and achieves an ideal distribution. Thus, for any set of arbitrary v functions that Φ_3 encounters, $z(\Phi_3^a , \varphi_3)$ represents the optimal evaluation. We now bound how far Φ_3 can be from this amount.

By definition, $z(\Phi_3^l , \varphi_3) = g(d^l) + \sum_i v^l(s_i)$. We know that $d^{l+c-\varepsilon} = d^l$ because $v^{l+c-\varepsilon}$ differs only by a constant from v^l at each slot. Thus, $z(\Phi_3^{l+c-\varepsilon} , \varphi_3) = z(\Phi_3^{l+c-\varepsilon} , \varphi_3) = g(d^l) + \sum_i v^{l+c-\varepsilon}(s_i) = g(d^l) + \sum_i v^l(s_i) + (c-\varepsilon)n$. This implies that $z(\Phi_3^l, \varphi_3) + (c - \varepsilon)n = z(\Phi_3^{l+c-\varepsilon}, \varphi_3)$; it remains to show that $z(\Phi_3^{l+c-\varepsilon}, \varphi_3) \geq z(\Phi_3^a, \varphi_3)$. Note that $z(\Phi_3^{l+c-\varepsilon}, \varphi_3) \geq z(\Phi_3^{l+c-\varepsilon}, \varphi_3)$ by definition of d^l. Also, $z(\Phi_3^{l+c-\varepsilon}, \varphi_3) \geq z(\Phi_3^a, \varphi_3)$ because $v^{l+c-\varepsilon}$ is an upper bound on each of the v's in the case of Φ_3^a and $g(d^a)$ is common to both terms. Thus $z(\Phi_3^l, \varphi_3) + (c - \varepsilon)n = z(\Phi_3^{l+c-\varepsilon}, \varphi_3) \geq z(\Phi_3^a, \varphi_3)$. Now we return to the real Φ_3. The optimal distribution is d^l, and in φ_3 the network extracts ε less revenue from each agent than Φ_3^l did because it does not set $\varepsilon = 0$. Thus, $z(\Phi_3, \varphi_3) + n\varepsilon = z(\Phi_3^l, \varphi_3)$. Combining the last two equations, we can conclude: $z(\Phi_3, \varphi_3) + cn \geq z(\Phi_3^a, \varphi_3)$, and thus that φ_3 is c-optimal. ∎

It follows from this statement that if we revert back to the setting from Sections 2.3 and 2.4 (where $v^u(s) = v^l(s)$), the network will lose only ε in revenue from each agent. It is only the change to bounds on valuation functions that causes the weaker claims on optimality for this mechanism and the next.

Corollary 2.15 (Φ_3, φ_3) *is ε-optimal for $v^l = v^u$.*

Proof. This follows directly from the preceding theorem, because $v^l = v^u$ implies that $c = \varepsilon$.

2.5.2 Implementation Considerations

We observe that it may involve less overhead to assign single, persistent signals to agents if the game will be repeated many times. In this case, the collective reward mechanism may be used as above but without the signalling phase, and with each agent a_j who did not participate counted by d^+ as having participated in slot $\sigma(j)$. This allows φ_3 to hold in the case where signals are not assigned repeatedly with the penalty that φ_3 will only be c-optimal for $c = \max_s\big(v^u(s) - v^l(s)\big) + \varepsilon$ when all agents participate.

2.6 Discriminatory Mechanism

A disadvantage of the bulletin board mechanism is that it reimburses some agents at the end of the game rather than simply waiving their fees. This requires tracking individual agents' behavior and executing more financial transactions, both of which could be costly to the network. Also, the bulletin board mechanism has non-optimal equilibria. Finally, irrational agents can harm others in both the bulletin board and collective reward mechanisms. These problems are eliminated by the discriminatory mechanism, Φ_4, which makes use of agent signals and also discriminates by offering different free slots to different agents (although, as we will see in Section 2.6.2 it makes new demands of the network that will sometimes be undesirable):

1. Each agent indicates that he will participate.
2. The network assigns signals to agents from $\{1, \ldots, t\}$ according to the d^* that is ideal for p^b.
3. The network chooses "potentially free" slots according to p^b.
4. Each agent indicates what slot he selects.
5. The network checks only those agents in each slot s_i that was picked to be "potentially free" (for all agents who chose other slots, $P_i = 0$) . If agent a_j in slot s_i has $\sigma(a_j) = s_i$ then $P_j = p^b(A_j)$; otherwise $P_j = 0$.

2.6.1 Equilibria

Agent a_i's dominant strategy is to choose the slot corresponding to his signal. The analysis exactly follows that for φ_3; we call this equilibrium φ_4. The only difference is that an agent's expected utility does not depend on other agents' strategies, and hence φ_4 is an equilibrium in dominant strategies. A consequence is that φ_4 is unique. By exactly the same argument that was given in the proof of theorem 2.13, (Φ_4, φ_4) is c-optimal for $c = \max_s\big(v^u(s) - v^l(s)\big)+\varepsilon$. The same corollary also holds, and so (Φ_4, φ_4) is ε-optimal for the special case where $v^u = v^l$.

It may seem disappointing from a game-theoretic point of view that neither strategy nor even payoffs under the discriminatory mechanism depend on the actions of other agents. However, this may be seen as an advantage of the discriminatory mechanism, since irrational agents are not able to harm others.

2.6.2 Implementation Considerations

As compared to the collective reward mechanism, the discriminatory mechanism makes two additional demands of the network. First, the network must keep track of the signals that are given to agents in the second step, so that they can be verified in the fifth step. In collective reward the system does not need any sort of user accounts; rather, it greedily assigns signals to agents, recording only the *number* of agents who received each signal.

Second, the discriminatory mechanism requires the network to verify user identities. In contrast, the collective reward mechanism simply counts the number of agents who chose each slot. Under the discriminatory mechanism the network only has to check the identity of agents from q slots on expectation, since agents who choose a slot that is not potentially free do not have to be checked. It would be possible for the network to assume that all agents in possibly free slots have played according to the dominant strategy and to randomly check only a subset of the agents in these slots, but this would reduce the penalty for defection and thus sacrifice c-optimality.

In order to permit this verification, the mechanism can assign signals to agents in two different ways. The obvious option is to assign signals to agents as described in theorem 2.11, to store the numbers in some sort of user account requiring login and then to verify that agents selected the appropriate slot by requiring them to log in again before using the network resource. This approach requires further data storage by the mechanism, but the resulting d will be ideal and thus the mechanism will be c-optimal as argued above. If this data storage is not desirable, a deterministic function may be used to calculate the slot that may be offered free to a given agent, and the same function may be used to determine whether each agent has selected the appropriate slot. For example, a hash of the agent's IP address—or of any other identifying information from the packet header—could be used. This approach has the disadvantage that it sacrifices optimality and for steps 2 and 5 in the mechanism, but the advantage that no information about identifying individual agents must be stored by the mechanism.[5] Indeed, if the function itself is publicized then the first two

[5]Another disadvantage is that an agent could register from one computer, receive a slot assignment, use the network from a second computer and be denied a chance for a free slot because the

steps may be omitted from the mechanism, requiring only one interaction between agents and the network.

2.7 Comparison of Different Mechanisms

Table 2.1 summarizes and contrasts the mechanisms discussed in this chapter. For convenience, we have divided the display into three parts: (i) a list of mechanism characteristics, (ii) a comparison of the outcomes of the mechanisms, and (iii) costs associated with executing the mechanisms.

2.8 Conclusions

Focused loading is a predictable network congestion problem. It is caused by a preference users have for transacting with a network resource at a specific time when the network charges transactions equally over a period of time. For example, focused loading frequently causes web servers to crash. In this chapter we have taken an economic approach to de-focusing load by devising incentive schemes for encouraging users to desynchronize their transaction times. While general congestion-management techniques may be applicable to this problem, the use of a specialized solution is attractive because additional information about the problem can be used to increase revenue and reduce demands on the network.

We present a theoretical model of the problem, and discuss four mechanisms that induce selfish agents to smooth out their resource demands by probabilistically waiving the cost of resource usage. We show one very simple mechanism that achieves a weak load-balancing equilibrium, and three other, somewhat more complex mechanisms that balance load in strict equilibria or dominant strategies. Two of our mechanisms concern the case where all agents have the same valuations for different time slots, and two generalize to the case where the mechanism knows only bounds

second computer's IP did not hash to the same signal. This could be addressed by requiring agents to use the network from the computer from which they registered, and permitting them to register again if they change their mind about which machine they want to use.

	Φ_1: **Preselection**	Φ_2: **Bulletin Board**	Φ_3: **Collective Reward**	Φ_4: **Discriminatory**
Earliest possible free slot selection	Before any time slots	After all time slots	After each time slot	After each time slot
Agent signals	No	No	Yes	Yes
The network must store agent signals	No	No	No	Yes, or hash IP
Agents may have different v functions	No	No	Yes	Yes
Time required for coordination phase	None	Substantial	Negligible	Negligible
Type of equilibrium or strategy	Weak equilibrium	Strict equilibrium	Strict equilibrium	Dominant strategy
Non-optimal equilibria exist	Yes	Yes	No	No
Revenue increases if agents deviate	No	Yes	Yes	Yes
Harmful collusion	No	No	No	No
Irrational actions harm other agents	No actions are irrational	Yes	Yes	No
Time cost after coordination phase	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Storage cost	$O(q)$ (free slots)	$O(t)$ (d)	$O(n)$ (moves)	$O(n)$ (signals, identities)
Communication cost	$O(n)$	$O(nt)$	$O(n)$	$O(n)$

Table 2.1: Comparison of Φ_1, Φ_2, Φ_3, Φ_4

on agent valuations. We prove optimality and ε-optimality of the revenue/load balancing trade-off in the first case, and a bound on the optimality of this trade-off in the second case.

Chapter 3

Bidding Rings in First Price Auctions

We identify a self-enforcing collusion protocol (a "bidding ring") for first-price auctions. Unlike previous work on this topic, we allow for the existence of multiple cartels in the auction, we include the choice of whether or not to collude as part of agents' strategy space, and we do not assume that non-colluding agents hold false beliefs. We show a Bayes-Nash equilibrium in which agents choose to join bidding rings when invited and truthfully declare their valuations to a ring center. Furthermore, we show that the existence of bidding rings benefits ring centers and all agents, both members and non-members of bidding rings, at the auctioneer's expense.

3.1 Introduction

We consider the question of how agents can gain by coordinating their bidding in non-repeated single-good auctions, even when all agents still act selfishly. The case of second-price auctions is well-studied; we concentrate on the comparatively less-studied case of first-price auctions. This problem is important for several reasons. First, collusion is a widespread phenomenon; understanding the topic theoretically can help auctioneers to modify the rules of their auctions in order to make collusion more difficult. Second, the problem of designing protocols that allow agents to maximize their expected utility in existing economic mechanisms is a topic of increasing study in artificial intelligence; *cf.*, [Wellman *et al.*, 2003]. Finally, there exist uses of bidding rings that might not fall under the legal definition of collusion. For example, a corporation could use a bidding ring to choose one of its departments to bid in an external auction, avoiding bidding against itself while avoiding dictatorship and respecting each department's self-interest. Collusion may also be permitted by the auctioneer: *e.g.*, by an internet market seeking to attract more bidders.

Graham and Marshall [1987] wrote one of the first formal papers on collusion, considering second-price auctions. This paper introduced the *knockout procedure*: agents announce their bids in a knockout auction; only the highest bidder goes to the auction but this bidder must pay a "ring center" the amount of his gain relative to the case where there was no collusion. The ring center pays each agent in advance; the amount of this payment is calculated so that on expectation the ring center will budget-balance *ex ante*, before knowing the agents' valuations.

Graham and Marshall's work has been extended to deal with variations in the knockout procedure, differential payments, and relations to the Shapley value [Graham *et al.*, 1990]. The case where only some of the agents are part of the cartel is discussed by Mailath and Zemsky [1991]. von Ungern-Sternberg [1988] discusses collusion in second-price auctions where the designated winner of a cartel is not the agent with the highest valuation. Although we are not aware of any work that presents this result, it is also easy to extend the protocol to an environment containing both multiple cartels and independent bidders.

Less formal discussion of collusion in auctions can be found in a wider variety of papers. For example, a survey paper that discusses mechanisms that are likely to facilitate collusion in auctions, as well as methods for the detection of such schemes, can be found in [Hendricks & Porter, 1989]. A discussion and comparison of the stability of rings associated with classical auctions can be found in [Robinson, 1985], concentrating on the case where the valuations of agents in the cartel are honestly reported. Collusion is also discussed in other settings, *e.g.*, aiming to influence purchaser behavior in a repeated procurement setting [Feinstein *et al.*, 1985] and in the context of general Bertrand or Cournot competition [Cramton & Palfrey, 1990].

3.1.1 Collusion in First-Price Auctions

A very influential paper by McAfee and McMillan [1992] presented the first theoretical results on collusion in first-price auctions. This work assumes that a fixed number of agents participate in the auction and that all agents are part of a single cartel that coordinates its behavior in the auction. The authors show optimal collusion protocols for “weak” cartels (in which transfers between agents are not permitted: all bidders bid the reserve price, using the auctioneer’s tie-breaking rule to randomly select a winner) and for “strong” cartels (the cartel holds a knockout auction, the winner of which bids the reserve price in the main auction while all other bidders sit out; the winner distributes some of his gains to other cartel members through side payments). A small part of the paper deals with the case where in addition to a single cartel there are also additional agents. However, results are shown only for two cases: (1) where non-cartel members bid without taking the existence of a cartel into account (*i.e.*, either they are irrational or they hold the false belief that no cartels exist) and (2) where each agent i has valuation $v_i \in \{0, 1\}$. The authors explain that they do not attempt to deal with general strategic behavior in the case where the cartel consists of only a subset of the agents; furthermore, they do not consider the case where multiple cartels can operate in the same auction.

3.1.2 Overview

Our goal is to extend past work on collusion in first-price auctions by relaxing many of the assumptions described above. First, we want to allow for the possibility that some bidders will not belong to a cartel, while preserving the standard game-theoretic setting in which agents are rational and do not hold false beliefs in equilibrium. Second, we want to allow for the possibility that more than one cartel will exist, introducing the new wrinkle that cartel members must reason about the behavior of other cartels. Other features of our model are that bidders have uncertainty about the number of participants in the auction, bidders’ valuations are real numbers drawn from an interval according to an arbitrary distribution (as compared, *e.g.*, to the case described above where valuations take one of only two discrete values), and the decision of whether or not to join a bidding ring is part of an agent's choice of strategy.

Section 3.2 describes what we consider to be the simplest economic environment and first-price auction mechanism that do not implicitly rule out the possibility of collusion. Section 3.3 gives two important lemmas that are necessary for our main results, but which do not directly concern collusion. Section 3.4 describes all our results concerning bidding rings. We extend the model from Section 3.2 to give a model of the economic environment and an augmented auction mechanism for bidding ring invitees. We show that this protocol results in a Bayes-Nash equilibrium, and also that the bidding ring protocol helps both members and non-members in several senses.

3.2 Modeling First-Price Auctions

In this section we will present an "auction hall" setting, in which bidders are aware of the number of other bidders placing bids in the auction hall, but are uncertain about the number of bidders who remained at home after losing a knockout auction. First, however, we introduce notation, and then present two simpler first-price auction models. We present results about these models that we will need throughout the remainder of the paper, and also argue why these models are insufficient for the study of collusion.

3.2.1 Auction Setting

An economic environment E consists of a finite set of agents who have non-negative valuations for a good at auction, and a distinguished agent 0—the seller or center. Denote the economic environment described here as E_c. Let $\mathcal{T}$ be the set of possible agent types. The type $\tau_i \in \mathcal{T}$ of agent i is the pair $(v_i, s_i) \in V \times \mathcal{S}$. v_i denotes an agent's valuation: his maximal willingness to pay for the good offered by the center. We assume that v_i represents a purely private valuation for the good, and that v_i is selected independently from the other v_j's of other agents from a known cumulative distribution, F, having density function f. We assume only that f is continuous, atomless, and has support on the interval $[0, 1]$. By s_i we denote agent i's signal: his private information about the number of agents in the auction. The set of possible signals $\mathcal{S}$ will be varied throughout the chapter. Let φ denote the null signal, and in E_c let $\mathcal{S} = \{\varphi\}$. Note, however, that the economic environment itself is always common knowledge, and so agents always have some information about the number of agents even when they always receive the null signal.

By $p_{s_i}(n)$ we denote the probability that agent i assigns to there being exactly n agents in the auction, conditioned on his signal s_i. We denote the whole distribution conditioned on i's signal as p_{s_i}. The utility function of agent i, $u_i : \mathbb{R} \to \mathbb{R}$ is linear, normalized with $u_i(0) = 0$. The utility of agent i (having valuation v_i) when asked to pay t is $v_i - t$ if i is allocated a good and $-t$ otherwise. Thus, we assume that there are no externalities in agents' valuations and that agents are risk-neutral. $b_i : \mathcal{T} \to \mathbb{R}^+ \cup \{0\}$ denotes agent i's strategy, a mapping from i's type τ_i to his declaration in the auction. This may be the null declaration 0, indicating that i will not participate in the auction.

3.2.2 Classical First-Price Auctions

The choice of information structure is very important for the study of collusion in first-price auctions. The most familiar case gives rise to what we will call the "classical" first-price auction, where the number of participants[1] is part of the economic environment (this is what we have called E_c). Using standard equilibrium analysis (*e.g.*, following Riley and Samuelson [1981]) a unique symmetric equilibrium can be identified:

Proposition 3.1 *If valuations are selected from a continuous distribution F then it is a unique symmetric equilibrium for each agent i to follow the strategy*

[1]When we say that n agents participate in the auction we do not count the distinguished agent 0, who is always present.

$$b(v_i) = v_i - F(v_i)^{-(n-1)} \int_0^{v_i} F(u)^{n-1} du.$$

Observe that the strategy is parameterized by valuation, but also depends on information from the economic environment. It will be notationally useful for us to be able to specify the amount of the equilibrium bid as a function of both v and n:

$$b^e(v_i, n) = v_i - F(v_i)^{-(n-1)} \int_0^{v_i} F(u)^{n-1} du. \tag{3.1}$$

We are interested in constructing a collusive agreement that requires low bidders to drop out of the main auction. It is obvious that such collusion is nonsensical in a classical first-price auction. When bidders' strategies depend on the number of agents in the economic environment, it makes no difference if cartel members with low valuations fail to submit bids. This is a problem with our auction model rather than with collusion in first-price auctions *per se*—in practice bidders might *not* know the exact number of agents in the economic environment.

3.2.3 First-Price Auctions with a Stochastic Number of Bidders

One way of modelling agents' uncertainty about the number of opponents they face is to say that the number of participants is drawn from a probability distribution; while the actual number of participants is not observed, the *distribution* is commonly known [McAfee & McMillan, 1987]. This setting requires that we redefine the economic environment; denote the new economic environment as E_s. Let the definition of agents in E_s be the same as in E_c. Denote the probability that n agents will participate in the auction as $p(n)$; let the support of p be any subset of $\{2, 3, \ldots\}$. We assume that after the number of agents is determined, the name of each agent is selected from the uniform distribution on $[0, 1]$. An equilibrium was demonstrated by Harstad *et al.* [1990]:

Proposition 3.2 *If valuations are selected from a continuous distribution F and the number of bidders is selected from the distribution p then it is a unique symmetric* *equilibrium for each agent i to follow the strategy*

$$b(v_i) = \sum_{j=2}^{\infty} \frac{F^{j-1}(v_i)p(j)}{\sum_{k=2}^{\infty} F^{k-1}(v_i)p(k)} \, b^e(v_i, j).$$

Observe that $b^e(v_i, j)$ is the amount of the equilibrium bid for a bidder with valuation v_i in a setting with j bidders as described in Section 3.2.2 above. p is deduced from the economic environment. We overload our previous notation for the equilibrium bid, this time as a function of the agent's valuation and the probability distribution p:

$$b^e(v_i, p) = \sum_{j=2}^{\infty} \frac{F^{j-1}(v_i)p(j)}{\sum_{k=2}^{\infty} F^{k-1}(v_i)p(k)} \, b^e(v_i, j). \tag{3.2}$$

This auction model is still unable to model collusion in a first-price auction. If each agent knows only the distribution of agents interested in participating in the auction, his strategy will not be affected if one or more interested agents drop out! Again, this is a deficiency with our model. In some settings agents may know how many agents bid in the auction, even though they may not know the number of agents who chose not to bid. For example, when an auction takes place in an auction hall, no bidder knows how many potential bidders stayed home, but every bidder can count the number of people in the room before placing his or her bid. It is in this sort of auction that collusion based on dropping agents with low valuations could work. We must first introduce a new type of auction to model this auction hall scenario.

3.2.4 First-Price Auctions with Participation Revelation

We define first-price auctions with participation revelation as follows:

1. Agents indicate their intention to bid in the auction.
2. The auctioneer announces n, the number of agents who registered in the first phase.
3. Agents submit bids to the auctioneer. The auctioneer will only accept bids from agents who registered in the first phase.

4. The agent who submitted the highest bid is awarded the good for the amount of his bid; all other agents are made to pay 0.

When a first-price auction with participation revelation operates in E_s, the equilibrium of the corresponding *classical* first-price auction holds.

Proposition 3.3 *In E_s it is an equilibrium of the first-price auction with participation revelation for every agent i to indicate the intention to participate and to bid according to $b^e(v_i, n)$.*

Proof. Agents always gain by participating in first-price auctions when there is no participation fee. The only way to participate in this auction is to indicate the intention to participate in the first phase. Thus the number of agents announced by the auctioneer is equal to the total number of agents in the economic environment. From Proposition 3.1 it is best for agent i to bid $b^e(v_i, n)$ when it is common knowledge that the number of agents in the economic environment is n. ■

In practice, a first-price auction with participation revelation may often be a more realistic model than a classical first-price auction, since it covers cases where bidders do not know *a priori* the number of opponents they will face. However, when bidders are unable to collude there is no strategic difference between these two mechanisms, possibly explaining why the simpler classical model is commonly used. For the study of bidding rings, however, the difference between the mechanisms is profound—we are now able to look for a collusive equilibrium in which bidder strategies to depend only on the number of other agents who "show up" for the auction, without having to assume that bidders are irrational or hold false beliefs about the economic environment.

3.3 Some Technical Results

This section proves several novel technical results that are necessary for our results about bidding rings, all concerning economic environments in which bidding rings do not exist, but in which the number of bidders is stochastic. First, we discuss the existence of an equilibrium in a setting where agents receive asymmetric information about the number of participants, and are furthermore subject to asymmetric payment rules. Second, we prove a key result about how changes in the distribution over the number of participants in a first-price auction in E_s affect the amount of the equilibrium bid b^e.

3.3.1 Regular Asymmetric Auctions

We describe a class of auction mechanisms that are asymmetric in the sense that all agents are subject to the same allocation rule but may receive different signals about the number of participants in the auction and may be made to pay according to different payment rules.[2] Let $S = \{\varphi, 2, 3, \ldots\}$. A bid from agent i is denoted as $\mu_i \in \mathbb{R}^+ \cup \{0\}$, the tuple of bids from all agents is denoted as $\pi \in \Pi$ and an auction's transfer function for agent i (determining i's payment) is $t_i : \mathbb{R}^+ \cup \{0\} \times \Pi \to \mathbb{R}$.

Definition 3.4 *An auction M_s is* aligned *with signal $s \in S$ if M_s allocates the good to an agent i with $\mu_i \in \max_j \mu_j$, and M_s is a symmetric truth-revealing direct mechanism for a stochastic number of agents drawn from p_s, each of whom receives the signal φ.*

By the revelation principle, for every distribution over the number of agents p and for every signal $s \in S$ there exists an aligned auction M_s.

Definition 3.5 *$\bar{M}$ is a* regular asymmetric auction *if it allocates the good to an agent i with $\mu_i \in \max_j \mu_j$, and if each agent i is made to transfer $t_{s_i}(\mu_i, \pi)$ to the center, with t_{s_i} taken from an auction M_{s_i} that is aligned with signal s_i.*

Lemma 3.6 *Truth-revelation is an equilibrium of regular asymmetric auctions.*

Proof. The payoff of agent i is uniquely determined by the allocation rule, the transfer function t_{s_i}, the distribution over the number of agents in the auction, and all agents' strategies. Assume that the other agents are truth revealing, then each other agent's behavior, the allocation rule, and agent i's payment rule are all identical in

[2]Of course, this is in addition to the sense in which *all* auctions are asymmetric: agents have different valuations for the good.

$\bar{M}$ and M_{s_i}. Conditioned on his private information s_i, agent i's posterior is that p_{s_i} is the distribution over the number of agents in the auction. Since truth-revelation is an equilibrium in M_{s_i} when the distribution of agents is p_{s_i}, truth-revelation is agent i's best response in $\bar{M}$. ■

3.3.2 Relating p to b^e

It is intuitive to expect that in first-price auctions, the amounts of agents' equilibrium bids increase with the number of participating agents. We can easily verify that this is true in the classical case:

Lemma 3.7 *$\forall v$, $\forall j \geq 2$, $b^e(v, j+1) > b^e(v, j)$.*

Proof. From Equation (3.1), we can write

$$b^e(v, j+1) - b^e(v, j) = \int_0^v \left(1 - \left(\frac{F(u)}{F(v)}\right)\right)\left(\frac{F(u)}{F(v)}\right)^{j-1} du. \tag{3.3}$$

The first factor in the integrand is clearly always positive, so the right-hand side of Equation (3.3) is positive. Thus $b^e(v, j)$ is strictly increasing in j. ■

This intuition does not transfer to first-price auctions with a stochastic number of bidders, in the sense that auctions with larger expected numbers of participants do not always yield higher equilibrium bids.

Example 3.8 *Consider a distribution p such that the probability mass is evenly divided between two numbers of agents, j_{low} and j_{high}, such that $j_{high} = j_{low} + \kappa$, and consider the strategy of agent i. The classical case is recovered if $\kappa = 0$, in which case i's equilibrium bid will just be $b^e(v_i, j_{low})$. If κ is increased to 1, the equilibrium bid increases by a finite amount to some $b^e(v_i, p) \in (b^e(v_i, j_{low}), b^e(v_i, j_{high}))$, as determined by Equation (3.2). As κ is increased to an arbitrarily high value, $F(v_i)^{j_{high}-1}$, the probability that agent i has the highest valuation when there are j_{high} agents involved, approaches zero. With arbitrarily close to unit probability, there will be j_{low} agents involved when agent i has the highest valuation, and Equation (3.2) indicates that i's* *bid will be arbitrarily close to the $\kappa = 0$ result. Thus while the $\kappa \to \infty$ distribution has a higher expected number of participants than the $\kappa = 1$ distribution, it elicits a lower equilibrium bid.*

This phenomenon also occurs among distributions of practical interest. So in a first-price auction with a stochastic number of participants, simply knowing that distribution p has a smaller expected number of participants than distribution p' is not enough to know which distribution gives rise to a lower symmetric equilibrium bid for a given valuation. In what follows, we identify a class of pairs of distributions (p, p') for which it does hold that $b^e(v_i, p) < b^e(v_i, p')$.

Let $r_j\left(F, v_i, p\right)$ denote the probability that j agents participate conditional on agent i having the highest valuation. This is equal to the probability that j agents participate and agent i has the highest valuation among these agents, normalized by the unconditional probability that agent i has the highest valuation. Let $Z\left(F, v_i, p\right)$ be the probability that agent i has the highest valuation given that his valuation is v_i.[3] Thus

$$Z\left(F, v_i, p\right) \equiv \sum_{k=2}^{\infty} F\left(v_i\right)^{k-1} p\left(k\right) \tag{3.4}$$

$$r_j\left(F, v_i, p\right) \equiv \frac{F\left(v_i\right)^{j-1} p\left(j\right)}{Z\left(F, v_i, p\right)}. \tag{3.5}$$

Observe that Equation (3.2) for the equilibrium bid in a stochastic first-price auction can be written in terms of the distribution $r(F, v_i, p)$:

$$b^e(v_i, p) = \sum_{j=2}^{\infty} r_j\left(F, v_i, p\right) b^e(v_i, j). \tag{3.6}$$

[3]We can use 2 rather than $-\infty$ as the lower limit of the sum in Equation (3.4) because $p(k)$ has support which is a subset of $\{2, 3, \ldots\}$. While $Z\left(F, v_i, p\right)$ is undefined when $F\left(v_i\right) = 0$, this technicality is of no practical interest.

The cumulative distribution $R_m(F, v_i, p)$ for the distribution r, denoting the probability that m or fewer agents participate conditional on i having the highest valuation, is given by

$$R_m(F, v_i, p) \equiv \sum_{j=2}^{m} r_j(F, v_i, p). \tag{3.7}$$

Let $\mathcal{D}_\ell$ be the set of all distributions $d : \mathbb{Z} \to \mathbb{R}$ with support on a subset of the integers greater than or equal to ℓ. Since auctions with less than two participating agents are not interesting, we are primarily concerned with distributions of numbers of agents that belong to $\mathcal{D}_2$. Let x and y be independent random variables respectively distributed according to $p \in \mathcal{D}_2$ and $q \in \mathcal{D}_0$, and consider the distribution of their sum, which we call p'. Since x and y are independent, the probability of their sum being m is just the sum of the product of the individual probabilities of values of x and y that sum to m:

$$p'(m) = \sum_{j=0}^{\infty} p(m-j)\, q(j). \tag{3.8}$$

Summing independent distributions in this way corresponds to convolution, which we denote symbolically as $p' = p * q$. Observe that convolution is associative and commutative. We denote repeated convolution of an arbitrary distribution d

$$\bigotimes_n d \equiv \overbrace{d * d * d * \ldots * d}^{d \text{ repeated } n \text{ times}}. \tag{3.9}$$

We define the Kronecker delta (an indicator function) as

$$\delta_m(j) = \begin{cases} 1 \text{ if } j = m; \\ 0 \text{ otherwise.} \end{cases} \tag{3.10}$$

We make use of the following identity, which can be inferred from Equation (3.8), later in the chapter:

$$\bigotimes_j \delta_k = \delta_{(j\cdot k)} \tag{3.11}$$

Lemma 3.9 *$\forall p, p' \in \mathcal{D}_2$, $\forall q \in \mathcal{D}_0$, $p' = p * q$ and $q(0) < 1$ implies that $b^e(v_i, p) < b^e(v_i, p')$.*

Proof. The proof has two parts. First we show that for every j, the probability that no more than j bidders participate conditional on bidder i having the highest valuation is at least as high when the number of agents is drawn from p as when it is drawn from p', and that for some j this probability is higher in p than in p'. Next, we show that this relationship between conditional probabilities implies that the equilibrium bid is smaller under p than under p'.

Step 1: First, we show that $\forall j$, $R_j(F, v_i, p') \leq R_j(F, v_i, p)$, and that $\exists j$, $R_j(F, v_i, p') < R_j(F, v_i, p)$.

Consider the difference between the cumulative distributions:

$$\begin{aligned}\Delta R_j &\equiv R_j(F, v_i, p) - R_j(F, v_i, p') \\ &= \sum_{m=-\infty}^{j} \left(\frac{F(v_i)^{m-1} p(m)}{Z(F, v_i, p)} - \frac{F(v_i)^{m-1} p'(m)}{Z(F, v_i, p')} \right).\end{aligned} \tag{3.12}$$

The denominators can be related as follows:

$$\begin{aligned} Z(F, v_i, p') &= \sum_{k=-\infty}^{\infty} F(v_i)^{k-1} \sum_{j=0}^{\infty} p(k-j)q(j) \\ &= \sum_{j=0}^{\infty} \sum_{k=-\infty}^{\infty} \left(F(v_i)F(v_i)^{j-1}F(v_i)^{k-j-1} \right) p(k-j)q(j) \qquad (3.13) \\ &= F(v_i) \sum_{j=0}^{\infty} F(v_i)^{j-1} q(j) \sum_{k=-\infty}^{\infty} F(v_i)^{k-j-1} p(k-j) \\ &= F(v_i)\, Z(F, v_i, q)\, Z(F, v_i, p). \qquad (3.14) \end{aligned}$$

Substituting (3.14) into Equation (3.12), and making use of Equation (3.8),

$$
\begin{aligned}
\Delta R_j &= \frac{1}{Z(F,v_i,p')} \sum_{m=-\infty}^{j} \Bigg(Z(F,v_i,q)\, F(v_i)^m\, p(m) \\
&\quad - F(v_i)^{m-1} \sum_{k=0}^{\infty} p(m-k)\, q(k) \Bigg) \qquad (3.15) \\
&= \frac{F(v_i)}{Z(F,v_i,p')} \Bigg(\sum_{k=0}^{\infty} q(k)\, F(v_i)^{k-1} \sum_{m=-\infty}^{j} F(v_i)^{m-1}\, p(m) \\
&\quad - \sum_{k=0}^{\infty} q(k)\, F(v_i)^{k-1} \sum_{m=-\infty}^{j} F(v_i)^{m-k-1}\, p(m-k) \Bigg) \qquad (3.16) \\
&= \frac{F(v_i)}{Z(F,v_i,p')} \sum_{k=0}^{\infty} q(k)\, F(v_i)^{k-1} \Bigg(\sum_{m=-\infty}^{j} F(v_i)^{m-1}\, p(m) \\
&\quad - \sum_{m=-\infty}^{j-k} F(v_i)^{m-1}\, p(m) \Bigg). \qquad (3.17)
\end{aligned}
$$

To obtain Equation (3.16), we have reordered the sums, made use of Equation (3.4) and performed factoring like that done to obtain Equation (3.13). To obtain Equation (3.17), we have factored the bracketed expression in (3.16) and shifted the dummy indices of the second sum.

When $k = 0$, the bracketed expression in Equation (3.17) is zero, so that term can be dropped from the sum. The bracketed sums can then be combined, yielding

$$
\Delta R_j = \frac{F(v_i)}{Z(F,v_i,p')} \sum_{k=1}^{\infty} \left(F(v_i)^{k-1}\, q(k) \sum_{m=j-k+1}^{j} F(v_i)^{m-1}\, p(m) \right). \qquad (3.18)
$$

Since $k \in [1, \infty)$ in Equation (3.18), the lower summand of the second sum is always less than or equal to the upper summand, so that sum is well-defined. Furthermore, all of the factors in Equation (3.18) are non-negative, so it remains only to be established whether $\Delta R_j > 0$ or $\Delta R_j = 0$. Since $p \in \mathcal{D}_2$, there exists some least element in the support of p; call this value m^*. For values of $j < m^*$ the second sum in Equation (3.18) gives exactly 0, and $\Delta R_j = 0$. Similarly, for all values of $j \geq m^*$, the second sum is nonzero, and since by assumption $\exists k > 0$ such that $q(k) > 0$, we have that $\Delta R_j > 0$. Thus for all j, $R_j(F, v_i, p') \leq R_j(F, v_i, p)$, and for some j, $R_j(F, v_i, p') < R_j(F, v_i, p)$.

Step 2: Now it must be established that ($\forall j, R_j(F, v_i, p') \leq R_j(F, v_i, p)$ and $\exists j$, $R_j(F, v_i, p') < R_j(F, v_i, p)$) implies $b^e(v_i, p) < b^e(v_i, p')$.

We must show that $\Delta b > 0$, where we use Equation (3.6) to write

$$\begin{aligned}\Delta b &\equiv b^e(v_i, p') - b^e(v_i, p) \\ &= \sum_{m=2}^{\infty} \left(r_m(F, v_i, p') - r_m(F, v_i, p)\right) b^e(v_i, m).\end{aligned}$$

We rewrite this sum using summation by parts (the discrete analog of integration by parts). This yields

$$\Delta b = \sum_{m=2}^{\infty} \left(b^e(v_i, m+1) - b^e(v_i, m)\right) \sum_{j=2}^{m} \left(r_j(F, v_i, p) - r_j(F, v_i, p')\right) \quad (3.19)$$

$$= \sum_{m=2}^{\infty} \left(b^e(v_i, m+1) - b^e(v_i, m)\right)\left(R_m(F, v_i, p) - R_m(F, v_i, p')\right). \quad (3.20)$$

To obtain Equation (3.19), we have also used the fact that both $r(F, v_i, p)$ and $r(F, v_i, p)$ are normalized. Lemma 3.7 tells us that $b^e(v_i, m)$ is strictly increasing in m; clearly it is always positive. Thus $b^e(v_i, m+1) - b^e(v_i, m) > 0\ \forall m$. Furthermore, from Step 1, $R_m(F, v_i, p) - R_m(F, v_i, p')$ is non-negative, and for all $m \geq m^*$ it is greater than zero. The right-hand side of Equation (3.20) is therefore a sum of products of non-negative factors, of which at least one is a product of strictly positive factors. Thus $\Delta b > 0$, or $b^e(v_i, p) < b^e(v_i, p')$. ∎

3.4 Bidding Rings for First-Price Auctions

This section contains the chapter's main technical results. We begin by extending the economic environment E_s to include the characteristics necessary for a model of bidding rings. We then give the bidding ring protocol for first-price auctions, based on a first-price auction with participation revelation as described in Section 3.2.4. We show an equilibrium of this auction, and demonstrate that both ring centers and agents gain under this equilibrium.

3.4.1 Bidding Ring Economic Environment

Our aim is not to model a situation where agents' *decisions* to collude are exogenous, as this would gloss over the question of whether the collusion is stable. We thus include the collusive protocol as part of the model and show that it is individually rational *ex-post* (*i.e.*, after agents have observed their valuations) for agents to choose to collude. However, we do consider exogenous the selection of the set of agents who are *offered* the opportunity to collude. Furthermore, we want to show the impact of the possibility of collusion upon non-colluding agents; indeed, even those agents who do collude must take into account the possibility that *other* groups of agents in the auction may also collude. We extend the economic environment E_s to consist of the distinguished agent 0, a randomly-chosen set of agents who have non-negative valuations for a good at auction and a set of ring centers who do not value the good, but may invite agents to participate in a bidding ring. We denote the new economic environment E_{br}.

Ring centers

Ring centers are not free to choose their own strategies; rather, they act as part of the mechanism for a subset of the agents in the economic environment. The selection of ring centers is similar to the selection of agents in the economic environment E_s. In this case, however, this finite set is considered a set of "potential ring centers". In Section 3.4.1 we will describe which potential ring centers are "actualized," *i.e.*, correspond to actual ring centers. We denote the probability that an auction will involve

n_c potential ring centers as $\gamma_C(n_c)$. γ_C may be any distribution the support of which is a subset of $\{2, 3, \ldots\}$. We assume that after the number of potential ring centers is determined, the name of each potential ring center is selected from the uniform distribution on $[0, 1]$.

Agents

The probability that n agents will be associated with a potential ring center is denoted $\gamma_A(n)$. γ_A may be any distribution the support of which is a subset of $\{1, \ldots\}$. If only one agent is associated with a potential ring center, the potential ring center will not be actualized and hence the agent will not belong to a bidding ring. In this way we model agents who participate directly in the auction without being associated with a ring center. If more than one agent is associated with a potential ring center, the ring center *is* actualized and all the agents receive an invitation to participate in the bidding ring. As before, we assume that after the number of agents is determined, the name of each agent associated with a potential ring center is selected from the uniform distribution on $[0, 1]$. The key consequence of our technical construction of ring center and agent names is that an agent's knowledge of the ring center with whom he is associated does not give him additional information about what other agents have been selected. Any other technique for providing this property may also be used; *e.g.*, constructions draw ring center and agent names from finite sets.

We can now give an expression for p, the distribution over the number of agents in the economic environment E_{br}:

$$p = \sum_{n=2}^{\infty} \gamma_C(n) \left(\bigotimes_n \gamma_A \right). \tag{3.21}$$

Types and Signals

Recall that the type $\tau_i \in \mathcal{T}$ of agent i is the pair $(v_i, s_i) \in V \times \mathcal{S}$. As in E_c, let v_i denote an independent private value for the good, drawn from F. Let $\mathcal{S} \subseteq \mathbb{N} \setminus \{0\}$; $s_i \in \mathcal{S}$ denotes agent i's private information about the number of agents in his bidding ring. Of course, if this number is 1 then there is no ring center for the agent to deal with, and he simply participates in the main auction. Note also that agents are neither aware of the number of potential ring centers for their auction nor the number of actualized potential ring centers, though they are aware of both distributions.

Beliefs

Once an agent is selected, he computes a posterior distribution over the number of agents in the economic environment. Not all agents will have the same beliefs—agents who have been signaled that they belong to a bidding ring will expect a larger number of agents than singleton agents.

We denote by $p_{s_i}(m)$ the probability that there are a total of m agents in the auction, conditioned on agent i's observation of his own signal s_i:

$$p_{s_i} = \sum_{n=2}^{\infty} \gamma_C(n) \left(\bigotimes_{n-1} \gamma_A \right) * \delta_{s_i}. \tag{3.22}$$

Finally, we denote by $p_{n,s_i}(m)$ the probability that there are a total of m agents in the auction, conditioned on i's signal that there are s_i agents in his ring and the additional information that there are a total of n bidding rings and/or singleton bidders in the auction:

$$p_{n,s_i} = \left(\bigotimes_{n-1} \gamma_A \right) * \delta_{s_i}. \tag{3.23}$$

3.4.2 First-Price Auction Bidding Ring Protocol

Any number of ring centers may participate in an auction. However, we assume that there is only a single collusion protocol, and that this protocol is common knowledge. What follows is the protocol of a ring center who approaches k agents and who operates in conjunction with a first-price auction with participation revelation in the economic environment E_{br}.

1. Each agent i sends a message μ_i to the ring center.

2. If all k agents accept the invitation then the ring center drops all bidders except the bidder with the highest reported valuation, who we will denote as bidder h. For this bidder the ring center indicates the intention to bid in the main auction, and places a bid of $b^e(\mu_h, p_{n,1})$.

3. If $d > 0$ agents decline participation then the ring center indicates an intention to bid in the main auction on behalf of every agent who accepted the invitation to the bidding ring. For each bidder i, the ring center submits a bid of $b^e(\mu_i, p_{n-d,k})$, where n is the number of bidders announced by the auctioneer.[4]

4. The ring center pays each member a pre-determined payment $c_{n,k} \geq 0$ whenever all bidders participate in the ring, which is independent of the outcome of the auction and the amount each bidder bid, but which can depend on n and k.

5. If bidder h wins in the main auction, he is made to pay $b^e(\mu_h, p_{n,1})$ to the center and $b^e(\mu_h, p_{n,k}) - b^e(\mu_h, p_{n,1})$ to the ring center.

We are now ready to prove the main theorem of the chapter.

Theorem 3.10 *It is a Bayes-Nash equilibrium for all bidding ring members to choose to participate and to truthfully declare their valuations to their respective ring centers, and for all non-bidding ring members to participate in the main auction with a bid of* $b^e(v, p_{n,1})$.

Proof. This proof is divided into sections. First, we prove that each category of bidders is best off choosing to participate. Next, we present a one-stage revelation mechanism and prove that it is equivalent to the bidding ring protocol when all agents participate. Finally, we show that the given strategies are in equilibrium when agents participate.

Participation of non-ring bidder: When there is no participation fee, it is always rational for a bidder to participate in a first-price auction.

[4]Observe that in this case, the n announced by the auctioneer will include the d additional agents who deviate from the bidding ring.

Participation of ring bidder: Because there is no participation fee, all bidding ring members will participate in the auction, but must decide whether or not to accept their bidding ring invitations. Consider the case where $c_{n,k} = 0$; clearly $c_{n,k} > 0$ only increases agents' incentive to accept the invitation. Assume that all agents except for i join their respective rings and bid truthfully, and agent i must decide whether to join his ring and bid truthfully or to decline the invitation and bid freely. In this discussion let n represent the *true* number of bidding rings and singleton bidders in the economic environment (*i.e.*, the value realized from the distribution γ_c).

First, consider a different setting, which we denote ($\star$): a first-price auction with a stochastic number of participants in economic environment E_s, with the number of agents distributed according to p_{n,s_i}. In ($\star$) all bidders have the same information as i, and are subject to the same payment rule: from Proposition 3.2 it is a best response for i to bid $b^e(v_i, p_{n,s_i})$. Clearly i's expected utility is the same in ($\star$) and when participating honestly in his bidding ring, because both auctions allocate the good to the bidder who submits the highest bid, have the same distribution over the number of agents, and implement the same payment rule for i. Thus our goal in this part of the proof is to show that i's expected utility after rejecting his bidding ring invitation is less than his expected utility in the equilibrium of ($\star$).

Given that all other bidders participate in bidding rings and follow the protocol, if the bidding ring did not respond to i's deviation, there would exist some distributions p and signals s_i for which i would prefer to decline the invitation. (Taking into account his signal and once the auctioneer has made an announcement, i would know that the number of agents is distributed according to p_{n,s_i}; however, if he were to deviate then all agents would bid in the main auction as though the number of agents were distributed according to $p_{n+1,1}$. For certain values of p and s_i, i's expected loss from causing the auctioneer to announce one more participant is less than his expected gain from being able to bid freely and from not having to make a payment to the ring center if he wins.)

According to the protocol, however, the bidding ring *does* change its behavior in response to deviation. If i declines the invitation the ring center will send all the other members of the ring into the main auction, causing the auctioneer to announce

$n+k-1$ participants. As a result there will be s_i-1 bidders placing bids of $b^e(v, p_{n,s_i})$ and $n-1$ other bidders placing bids of $b^e(v, p_{n+s_i-1,1})$. We can show that these $n-1$ bidders will decrease i's expected utility by bidding too high. Recall Equation (3.23): $p_{n,s_i} = \left(\bigotimes_{n-1} \gamma_A\right) * \delta_{s_i}$, and so $p_{n+s_i-1,1} = \left(\bigotimes_{n+s_i-2} \gamma_A\right) * \delta_1$. We can write $\gamma_A = g_A * \delta_1$, where g_A is the distribution over the number of agents in a bidding ring beyond the first agent. Then

$$\begin{aligned} p_{n+s_i-1,1} &= \left(\bigotimes_{n-1} \gamma_A\right) * \left(\bigotimes_{s_i-1} \gamma_A\right) * \delta_1 \\ &= \left(\bigotimes_{n-1} \gamma_A\right) * \left(\left(\bigotimes_{s_i-1} g_A\right) * \delta_{s_i-1}\right) * \delta_1 \\ &= p_{n,s_i} * \left(\bigotimes_{s_i-1} g_A\right). \end{aligned} \tag{3.24}$$

Since γ_A has support on a subset of the positive integers, it follows that g_A has support on a subset of the integers greater than or equal to zero. And since $\gamma_A(1) < 1$, $g_A(0) < 1$. It then follows from Lemma 3.9 that $b^e(v, p_{n+s_i-1,1}) > b^e(v, p_{n,s_i})$. Thus the singleton bidders and other bidding rings will bid a higher[5] function of their valuations than the equilibrium amount in $(\star)$. It always reduces a bidder's expected gain in a first-price auction to cause other bidders to bid more, because it reduces the chance that he will win without affecting his payment if he does win. This is the effect of i declining the offer to join his bidding ring: the $k-1$ other bidders from i's bidding ring bid according to the equilibrium of $(\star)$, but the $n-1$ singleton and bidding ring bidders submit bids that exceed this amount. Therefore i's expected utility is smaller if he declines the offer to participate than if he accepts it.

One-Stage Mechanism: Define the one-stage mechanism M as follows:

1. The center announces n, the number of bidders in the main auction.

[5]Note that this occurs because the singleton bidders and other bidding rings in the main auction follow a strategy that depends on the number of bidders announced by the auctioneer; hence they bid as though all the $k-1$ bidders from the disbanded bidding ring might each be independent bidding rings.

2. Each bidder i submits a bid μ_i to the mechanism.

3. The bidder with the highest bid is allocated the good and is made to pay $b^e(\mu_i, p_{n,s_i})$.

4. All bidding ring members are paid c_{n,s_i}.

M has the same payment rule for bidding ring bidders as the bidding ring protocol given above, but no longer implements a first-price payment rule for singleton bidders. Observe that the original auction is efficient under the strategies stated in the theorem because each bidder i bids $b^e(v_i, p_{n,1})$ in the main auction. Thus, in order to prove that the strategies given in the statement of the theorem constitute an equilibrium, it is sufficient to show that truthful bidding is an equilibrium for all bidders under M.

Equilibrium: Assume that all other bidders bid truthfully, and consider the strategy of bidder i. This bidder's posterior distribution over the number of other bidders he faces, given his signal s_i and the auctioneer's announcement that there are n bidders in the main auction, is p_{n,s_i}. Since agent i is made to pay $b^e(\mu_i, p_{n,s_i})$ if he wins, and since the good is always allocated to the agent who submits the highest message, M is regular asymmetric. From Lemma 3.6, agent i's best response to truthful bidding in a regular asymmetric auction is to bid truthfully. Observe that this analysis holds for both non-ring and ring bidders since it makes no assumptions about s_i. If i is a ring bidder then he gets the additional payment c_{n,s_i}, but this payment does not depend on the amount of his bid, and so has no effect on his choice of how to bid given his decision to participate. ■

Note that this equilibrium gives rise to an economically-efficient allocation, as was mentioned in the proof of the theorem. The highest bidder in each bidding ring always bids in the main auction, and every bidder in the main auction places a bid according to the same function, which is monotonically increasing in the bidder's valuation.

The equilibrium from Theorem 3.10 is not unique. We can show another equilibrium, where no agents accept bidding ring invitations and agents bid according to the equilibrium for first-price auctions with participation revelation demonstrated in

Proposition 3.3.

Proposition 3.11 *It is a Bayes-Nash equilibrium for each bidding ring invitee to decline his bidding ring invitation, and for each agent i to bid $b^e(v_i, n)$.*

Proof. If at least one agent declines the invitation to join a bidding ring, other invitees of that bidding ring are no worse off if they decline as well. (If they decline then they can bid freely, rather than being made to submit bids of a particular form.) If no agents join bidding rings then agents' signals contain no useful information. Thus the argument from Proposition 3.3 applies, and it is a Bayes-Nash equilibrium for each bidder to submit a bid of $b^e(v_i, n)$. ∎

3.4.3 Are Bidding Rings Helpful?

First of all, we show that ring centers gain on expectation from running bidding rings.

Theorem 3.12 *The ring center gains on expectation if it pays agents $c_{n,k} = \frac{1}{k}(g_{n,k} - c'_{n,k})$ with $0 < c'_{n,k} \leq g_{n,k}$ and*

$$g_{n,k} = k \int_0^\infty f(v_i) \sum_{j=2}^{\infty} p_{n,k}(j) F^{j-1}(v_i) \left(b^e(v_i, p_{n,k}) - b^e(v_i, p_{n,1})\right) dv_i,$$

and is budget-balanced on expectation when $c'_{n,k} = 0$.

Proof. Since the distribution $p_{n,k}$ is just $p_{n,1}$ with $k-1$ singleton agents added, $p_{n,k} = p_{n,1} * \delta_{k-1}$. Since $k \geq 2$, it follows from Lemma 3.9 that $b^e(v_i, p_{n,k}) > b^e(v_i, p_{n,1})$. This proves that the ring center always receives a positive payment when a ring member wins. $g_{n,k}$ is the ring center's *ex ante* expected gain if all k invited agents behave according to the equilibrium in Theorem 3.10, the auctioneer announces n participants, and the ring center makes no payment to the agents. Thus the ring center will gain on expectation if each ring member's unconditional payment is less than $\frac{1}{k} g_{n,k}$, and will budget-balance on expectation when each ring member's payment is exactly $\frac{1}{k} g_{n,k}$. ∎

The payment of c to all bidders follows an idea from [Graham & Marshall, 1987] for returning a ring center's profits to bidders without changing incentives. In equilibrium the ring center will have an expected profit of $c'_{n,k}$, though it will lose $kc_{n,k}$ whenever the winner of the main auction does not belong to its ring. If a ring center wants to be guaranteed never to lose money, it can set $c'_{n,k} = g_{n,k}$.

There are several ways of asking whether *bidders* gain by being invited to join bidding rings. One natural question is whether bidders are better off being invited to a bidding ring or being sent to the auction as singleton bidders.

Theorem 3.13 *An agent i has higher expected utility in a bidding ring of size k bidding as described in Theorem 3.10 than he does if the bidding ring does not exist and k additional agents (including i) participate directly in the main auction as singleton bidders, again bidding as described in Theorem 3.10, for $c_{n,k} \geq 0$.*

Proof. Consider the counterfactual case where agent i's bidding ring does not exist, and all the members of this bidding ring are replaced by singleton bidders in the main auction. We show that i is better off as a member of the bidding ring (even when $c_{n,k} = 0$) than in this case. If there were n potential ring centers in the original auction and k agents in i's bidding ring, then the auctioneer would announce $n + k - 1$ as the number of participants in the new auction. In both cases the auction is economically efficient, which means i is better off in the auction that requires him to pay a smaller amount when he wins. Under the equilibrium from Theorem 3.10, as a singleton bidder i will bid $b^e(v_i, p_{n+k-1,1})$. If he belonged to the bidding ring and followed the same equilibrium i would bid $b^e(v_i, p_{n,k})$. As argued in the proof of Theorem 3.10, Lemma 3.9 shows that $\forall k \geq 2, \forall n \geq 2, \forall v, b^e(v, p_{n+k-1,1}) > b^e(v, p_{n,k})$, and so our result follows. ■

Intuitively, an agent gains by not having to consider the possibility that other bidders who would otherwise have belonged to his bidding ring might themselves be bidding rings. We can also show that singleton bidders and members of other bidding rings benefit from the existence of each bidding ring in the same sense. Following an argument similar to the one in Theorem 3.13, other bidders gain from not having to consider the possibility that additional bidders might represent bidding rings.

Paradoxically, as long as $c'_{n,k} > 0$, other bidders' gain from the existence of a given bidding ring is greater than the gain of that ring's members.

Corollary 3.14 *In the equilibrium described in Theorem 3.10, singleton bidders and members of other bidding rings have higher expected utility when $k \geq 2$ agents form a bidding ring than when k additional agents participate directly in the main auction as singleton bidders.*

Proof. Consider a singleton bidder in the first case, where the ring of k agents does exist. (It is sufficient to consider a singleton bidder, since other bidding rings bid in the same way as singleton bidders.) Following the equilibrium from Theorem 3.10 this agent would submit the bid $b^e(v_i, p_{n,1})$. Theorem 3.13 shows that it is better to belong to a bidding ring (and thus to bid $b^e(v_i, p_{n,k})$) than to be a singleton bidder in an auction with the same number of agents (and thus to bid $b^e(v_i, p_{n+k-1,1})$. From the argument in Theorem 3.12 we know that $b^e(v_i, p_{n,1}) < b^e(v_i, p_{n,k})$. Thus $\forall k \geq 2, b^e(v_i, p_{n,1}) < b^e(v_i, p_{n+k-1,1})$. ■

Another way of showing that bidding rings are helpful is to demonstrate that bidders prefer a world with bidding rings to a world without. We consider two other settings: an auction with participation revelation in economic environment E_s, and an auction with a stochastic number of bidders in E_s. First we compare the three environments *ex-post*, asking which an agent would prefer given knowledge of his own type. (Recall that we have defined an agent's type to include his signal s_i about the number of agents in the economic environment.) Second, we compare the environments *ex-ante*, asking which environment an agent would prefer if he knew the distribution over types but did not know what type he would receive.

Theorem 3.15 ***(ex-post ring)*** *For all $\tau_i \in \mathcal{T}$, for all $k \geq 2$, for all $n \geq 2$, for all $c_{n,k} > 0$, agent i obtains greater expected utility by:*

1. *participating in a bidding ring of size k in E_{br} and following the equilibrium from Theorem 3.10; than by*

2. *participating in a first-price auction with participation revelation in E_s with number of bidders distributed according to $p_{n,k}$; or by*

3. *participating in a first-price auction with a stochastic number of participants in E_s with number of bidders distributed according to $p_{n,k}$.*

When $c_{n,k} = 0$, agent i obtains the same expected utility in all three cases.

Proof. For an efficient first-price auction, an agent i's expected utility EU_i is $\sum_{j=2}^{\infty} p(j)F^{j-1}(v_i)b$, where $p(j)$ is the probability that there are a total of j agents in the economic environment, $F^{j-1}(v_i)$ is the probability that i has the high valuation among these j agents, and b is the amount of i's bid.

First, we consider case (1). Let $EU_{i,bc}$ denote agent i's expected utility in E_{br} as a member of a bidding ring of size k, in the equilibrium from Theorem 3.10. Recall that in this equilibrium the bidder with the globally highest valuation always wins, and if bidder i wins he will be made to pay $b^e(v_i, p_{n,k})$. In any case i will receive an unconditional positive payment of $c_{n,k}$.

$$EU_{i,bc} = \sum_{j=2}^{\infty} p_{n,k}(j)F^{j-1}(v_i)\left(v_i - b^e(v_i, p_{n,k})\right) + c_{n,k} \tag{3.25}$$

We now consider case (2). From Proposition 3.3 it is an equilibrium for agent i in economic environment E_s to bid $b^e(v_i, j)$ in a first-price auction with participation revelation, where j is the number of bidders announced by the auctioneer. Since the number of agents is distributed according to $p_{n,k}$, agent i's expected utility in a first-price auction with participation revelation, which we denote $EU_{i,pr}$, is

$$\begin{aligned}
EU_{i,pr} =& \sum_{j=2}^{\infty} p_{n,k}(j)F^{j-1}(v_i)\left(v_i - b^e(v_i, j)\right) && (3.26)\\
=& \frac{\sum_{\ell=2}^{\infty} p_{n,k}(\ell)F^{\ell-1}(v_i)}{\sum_{\ell'=2}^{\infty} p_{n,k}(\ell')F^{\ell'-1}(v_i)} \sum_{j=2}^{\infty} p_{n,k}(j)F^{j-1}(v_i)\left(v_i - b^e(v_i, j)\right) \\
=& \sum_{\ell=2}^{\infty} p_{n,k}(\ell)F^{\ell-1}(v_i)\left(v_i - \sum_{j=2}^{\infty} \frac{p_{n,k}(j)F^{j-1}(v_i)}{\sum_{\ell'=2}^{\infty} p_{n,k}(\ell')F^{\ell'-1}(v_i)} b^e(v_i, j)\right) \\
=& \sum_{\ell=2}^{\infty} p_{n,k}(\ell)F^{\ell-1}(v_i)\left(v_i - b^e(v_i, p_{n,k})\right). && (3.27)
\end{aligned}$$

Observe that we make use of the definition of $b^e(v_i, p)$ from Equation (3.2). Equation (3.27) is agent i's expected utility in case (3), so i's expected utility is equal in cases (2) and (3).

Intersecting equations (3.25) and (3.27), we get

$$EU_{i,bc} - EU_{i,pr} = c_{n,k}. \tag{3.28}$$

When $c_{n,k} > 0$, agent i's expected utility is strictly greater in case (1) than in cases (2) and (3); when $c_{n,k} = 0$ he has the same expected utility in all three cases. ∎

What about agents who do not belong to bidding rings? We can show in the same way that they are not harmed by the existence of bidding rings: they are neither better nor worse off in the bidding ring economic environment than facing the same distribution of opponents in the two cases described above.

Corollary 3.16 ***(ex-post singleton)*** *For all $\tau_i \in \mathcal{T}$, for all $n \geq 2$, agent i obtains the same expected utility by:*

1. *participating as a singleton bidder in E_{br} and following the equilibrium from Theorem 3.10; as by*
2. *participating in a first-price auction with participation revelation in E_s with number of bidders distributed according to $p_{n,1}$; and by*
3. *participating in a first-price auction with a stochastic number of participants in E_s with number of bidders distributed according to $p_{n,k}$.*

Proof. We follow the same argument as in Theorem 3.15, except that $k = 1$ and $EU_{i,bc}$ does not include $c_{n,k}$. Thus we get $EU_{i,bc} = EU_{i,pr}$. ∎

We now consider the *ex-ante* case. Observe that in this case an agent does not know whether or not he will be invited to a ring, as this is part of his type.

Corollary 3.17 ***(ex-ante)*** *For all $n \geq 2$, as long as $\exists n$, $\exists k$, $\gamma_c(n) > 0$ and $\gamma_a(k) > 0$ and $c_{n,k} > 0$, agent i obtains greater expected utility by:*

1. *participating in E_{br} and following the equilibrium from Theorem 3.10; than by*
2. *participating in a first-price auction with participation revelation in E_s with number of bidders distributed according to p; or by*
3. *participating in a first-price auction with a stochastic number of bidders in E_s with number of bidders distributed according to p.*

When $\forall n$, $\forall k$, $c_{n,k} = 0$, agent i obtains the same expected utility in both cases.

Proof. In case (1) agent i's expected utility (given an arbitrary n, which will be announced by the auctioneer rather than being a part of i's type) is

$$EU_{i,bc} = \int_0^\infty f(v_i) \sum_{k=1}^\infty \frac{k\gamma_A(k)}{\sum_{k'=1}^\infty k'\gamma_A(k')} \sum_{j=2}^\infty p_{n,k}(j) F^{j-1}(v_i) \\ (v_i - b^e(v_i, p_{n,k}))\, dv_i + c_{n,k} \quad (3.29)$$

In case (2) agent i's expected utility is

$$EU_{i,pr} = \int_0^\infty f(v_i) \sum_{k=1}^\infty \frac{k\gamma_A(k)}{\sum_{k'=1}^\infty k'\gamma_A(k')} \sum_{j=2}^\infty p_{n,k}(j) F^{j-1}(v_i)\,(v_i - b^e(v_i, j))\, dv_i \quad (3.30)$$

We can make the same argument as in Theorem 3.15, starting with equations (3.29) and (3.30) instead of equations (3.25) and (3.26). The only difference is that after intersecting the equations we conclude that i prefers case (1) as long as there exist a pair (n, k) which are possible (*i.e.*, $\gamma_c(n) > 0$ and $\gamma_a(k) > 0$) for which $c_{n,k} > 0$, and that otherwise i is indifferent between the three cases. ∎

3.4.4 Comparing Equilibria in E_{br}

The theorems and corollaries in Section 3.4.3 allow us to compare the equilibrium from Theorem 3.10 with the equilibrium from Proposition 3.11.

Corollary 3.18 *Ex-ante, all bidders prefer the equilibrium from Theorem 3.10 to the equilibrium from Proposition 3.11. Ex-post, bidding ring invitees prefer the equilibrium from Theorem 3.10 to the equilibrium from Proposition 3.11, and singleton bidders are indifferent between the equilibria.*

Proof. A bidder's expected utility under the equilibrium from Proposition 3.11 (in economic environment E_{br}) is the same as his expected utility from an auction with participation revelation in economic environment E_s with the same distribution over the number of bidders. (This is obvious from the proof of Proposition 3.11.) Thus the result is immediate from Theorem 3.15, Corollary 3.16 and Corollary 3.17. ■

Since both bidders (and, trivially, ring centers) prefer the equilibrium from Theorem 3.10 to the equilibrium from Proposition 3.11, it follows that auctioneers must have the opposite preference. It turns out that auctioneers can disrupt bidding rings by making a slight change to the rules of the auction, so that the strategies described in Theorem 3.10 are no longer in equilibrium but the equilibrium from Proposition 3.11 is preserved. This can be achieved by making it possible for bidders to place bids in both their bidding rings and the main auction without detection by the ring center, either by allowing bidders to bid under "false names," or by refraining from publicly disclosing the winner of the auction.[6]

If bidders can bid both in their bidding rings and anonymously in the main auction, the equilibrium from Theorem 3.10 breaks down in the following way. A bidder i can accept the invitation to join the bidding ring but place a very low bid with the ring center; at the same time, i can directly submit a competitive bid in the main auction. Agent i will gain by following this strategy when all other agents follow the strategies specified in Theorem 3.10 because accepting the invitation to join the bidding ring ensures that the ring does drop all but one of its members and also causes the high bidder to bid less than he would if he were not bound to the collusion protocol. If the bidding ring drops any bidders other than i then all agents' bids will also be lowered because the number of participants announced by the auctioneer will be smaller, compared to the case where the bidding ring did not exist or where it was disbanded.

[6]Observe that these options may not be available to all auctioneers; *e.g.*, government auctions may be prohibited from allowing false-name bidding and required to publicly disclose winners.

However, if false-name bidding is impossible and the winner of the auction is publicly disclosed then the ring center can detect an agent who has deviated in this way. Because the agent has agreed to participate in the bidding ring the ring center has the power to punish this agent and make the deviation unprofitable.

3.5 Discussion

3.5.1 Assumptions

To emphasize the generality of our work and to suggest directions in which it might be extended, we restate our substantive assumptions here:

- Bidders are risk-neutral, with independent private values drawn from an arbitrary continuous and atomless distribution f on the interval $[0, 1]$.
- The distribution over the number of agents in the economic environment has the property that the numbers of agents invited to each bidding ring are independent.
- Invitations to join bidding rings are exogenous, and each bidder receives at most one invitation.
- Bidders are unable to place bids both in the bidding ring and directly in the main auction.
- Only a single bidding ring protocol is used in the auction.

3.5.2 Conclusions

We have presented a formal model of bidding rings which in many ways extends models traditionally used in the study of collusion. Most importantly, all agents behave strategically and take into account the possibility that groups of other agents will collude. Other features of our setting include a stochastic number of agents and of bidding rings in each auction, and revelation by the auctioneer of the number of bids received. The strategy space is expanded so that the decision of whether or not to join a bidding ring is part of an agent's choice of strategy. Bidding rings make money on expectation, and can optionally be configured so they never lose money.

We showed a bidding ring protocol for first-price auctions that leads to a (globally) efficient allocation in equilibrium, and which can optionally avoid the use of side-payments. In this equilibrium all invited agents choose to participate, even when the bidding ring operates in a single auction as opposed to a sequence of auctions. This means that the protocol's stability does not rely on the threat of an agent being denied future opportunities to collude.

We asked the question of whether agents gain by participating in bidding rings in first-price auctions in three different ways:

1. Could any agent gain by deviating from the protocol?
2. Would any agent be better off if his bidding ring did not exist?
3. Would any agent would be better off (either *ex-post* or *ex-ante*) in an economic environment that did not include bidding rings at all?

We have shown that agents are strictly better off in all three senses. (In the third sense, the gain is only strict when ring centers make a side-payment to agents.) We have also shown that each bidding ring causes *non-members* to gain in the second sense, and does not hurt them in the third sense.

Chapter 4

Local-Effect Games

We present a new class of games, local-effect games (LEGs), which exploit structure in a different way from other compact game representations studied in AI. We show both theoretically and empirically that these games often (but not always) have pure-strategy Nash equilibria. Finding a potential function is a good technique for finding such equilibria. We give a necessary and sufficient conditions for LEGs to have potential functions, and provide the functions for each case. We also show a general case where pure-strategy equilibria exist in the absence of potential functions. In experiments, we show that myopic best-response dynamics converge quickly to pure strategy equilibria in games not covered by our positive theoretical results.

4.1 Introduction

Games have long been studied in AI as a model of both competitive and cooperative multiagent interactions. While many AI researchers have concentrated on cooperative settings (see, *e.g.*, [Boutilier, 1996; Guestrin *et al.*, 2001]), there has also been growing interest in formulating competitive settings as games and in computing Nash equilibria for these games (see *e.g.*, [Billings *et al.*, 2003; Reeves & Wellman, 2003; Conitzer & Sandholm, 2002]). The computational obstacles to computing equilibria in general games has led to a parallel line of work on compact representations of games with large numbers of players, and games for which the computation of equilibria is tractable [Koller & Milch, 2001; Vickrey & Koller, 2002; Kearns *et al.*, 2001; Kearns & Mansour, 2002; Roughgarden & Tardos, 2001; Mura, 2000]. Much work in this vein has been based on the exploitation of one of two kinds of locality. First, some approaches exploit unconditional independencies between players' abilities to affect each other's payoffs [Koller & Milch, 2001; Vickrey & Koller, 2002; Kearns *et al.*, 2001; Blum *et al.*, 2003]. Second, some approaches exploit symmetry in utility functions along with context-specific independencies between players' effects on each other; more precisely, cases in which players' abilities to affect each other depend on the actions they choose. Here we study games in this second class, because we believe that this sort of context-specific independence is more common in real-world games.

Although compact representation has not been a primary motivation for economists, some work from economics does fall into the framework defined above. Most influentially, *congestion games* were defined by Rosenthal [1973]. In these games each agent i selects a subset S_i of a set of resources R; where n_r is the number of agents who choose resource $r \in R$, and F_r are arbitrary functions for each r, agent i pays:

$$p_i(S_i, n) = \sum_{r \in S} F_r(n_r) \tag{4.1}$$

The sets of actions available to each agent do not have to be identical or even overlapping. Observe that agent payoffs often do not exhibit any unconditional independencies: whenever two agents have action choices that name the same resource, they can affect each other's payoffs. On the other hand, context-specific independence does exist when two agents choose non-intersecting resource subsets. The main result in [Rosenthal, 1973] was that congestion games always have pure-strategy Nash equilibria (PSNE) . This is important because, although all games have mixed-strategy Nash equilibria [Nash, 1950], there are relatively few known classes of interesting games with pure strategy equilibria. At the same time, pure strategy equilibria are attractive: they can be more likely to arise in play as they are more intuitive than mixed-strategy equilibria for many players; they can be easier for agents to coordinate to; as there are a finite number of pure strategy profiles in a given game, they can be easier to compute than mixed strategy equilibria.

Rosenthal's work was extended by Monderer and Shapley [1996], who showed that the class of congestion games is equivalent to the class of games with *potential functions*. A potential function $P : A_1 \times \cdots \times A_n \rightarrow \mathbb{R}$ maps agents' joint actions to a real number, with the property that if X and Y are joint actions differing only in the action choice of one agent i, $P(X) - P(Y)$ is equal to the difference in i's payoff from selecting the two actions. This result is useful because it means that the construction of a potential function is sufficient for showing the existence of a pure-strategy equilibrium. Potential functions can also be used to compute equilibria: the set of pure-strategy Nash equilibria is equivalent to the set of joint actions that maximize P.

Recent work in computer science and AI has explored classes of games inspired by and extending congestion games. For example, Kearns et al. examined games with bounded effects [Kearns & Mansour, 2002], and Roughgarden studied a nonatomic variant [Roughgarden & Tardos, 2001]. In this chapter we propose a new class, which we call local-effect games.

4.2 Local-Effect Games

In congestion games, whenever two agents affect each other's payoff, they each do so by the same amount. *Local-effect games* (LEGs) model situations where agents' effects on each other are potentially asymmetric. Generally, action A locally affects action B if the utility of agents taking action B depends on some function $F_{A,B}$ of the number of agents taking action A, but the utility of agents taking action A depends on a different function $F_{B,A}$ of the number of agents taking action B.

There are many natural settings which are modeled by such locally-effecting actions. One problem domain that has been studied in economics for three quarters of a century is the *location problem* [Hotelling, 1929]. These problems model situations where agents must choose a location to operate their business in the presence of other competing agents, and each agent's profit depends on how far she is from her competitors. The canonical example concerns ice cream vendors who must choose a spot on the beach to set up a kiosk, with agents' utility depending on how many other ice cream sellers have located themselves in the same or adjacent areas. Work from economics on this problem has usually dealt with one-dimensional continuous spaces and has not modeled local effects explicitly; also, game theoretic analyses have typically considered only settings with 2 or 3 agents (see *e.g.*, [Osborne & Pitchik, 1987]). It is easy to think of many variants on the location problem: ice cream sellers arranging themselves around a lake (ring structure); vendors opening coffee houses in a city (grid structure); pairs at a cocktail party trying to pick a quiet room, with noise proportional to the number of people in the room, and noise also emanating from nearby rooms (arbitrary structure).

Another natural domain modeled by LEGs is a *role formation game*, where agents can take on one of a set of partially-substitutable roles. Agents are rewarded according to the amount of work they do, so their payoff is reduced as other agents adopt the same or similar roles.

Formally, let $\langle G, \mathcal{F}, n\rangle$ be a local-effect game for n agents. G is a graph, defined as $G = \langle \mathcal{A}, E\rangle$. $\mathcal{A}$ is the set of nodes in the graph, which correspond to the actions available to each player in the game.[1] E is a set of pairs of nodes, where $(a, a') \in E$ denotes that a (directed) edge runs from a to a'. Let $neigh(i)$ denote the set of nodes from which there are edges that terminate at node i. The function $\mathcal{F}_{a,a'} : \mathbb{Z}^{+} \cup \{0\} \to \mathbb{R}$ labels every node and edge in G: each node a is labelled by a *node function* $\mathcal{F}_{a,a}$ and each edge from a to a' is labelled by an *edge function* $\mathcal{F}_{a,a'}$. (For notational convenience, we define $\mathcal{F}$ for *every* pair of nodes; $(a, a') \notin E$ implies $\forall x,\ \mathcal{F}_{a,a'}(x) = 0$.)

Let D denote the distribution of players across actions, and $D(a)$ denote the number of players who chose action $a \in \mathcal{A}$. We can now state the cost function of an agent a who chose action $i \in \mathcal{A}$:

$$cost(a) = \mathcal{F}_{i,i}(D(i)) + \sum_{j \in neigh(\mathcal{A})} \mathcal{F}_{j,i}(D(j)). \tag{4.2}$$

[1]We note in passing that a natural extension to local-effect games is the case where the set of actions available to each agent i is $A_i \subseteq \mathcal{A}$.

We denote a *class* of local-effect games by the pair $\langle G, \mathcal{F} \rangle$: *i.e.*, the set of local-effect games having the same graph, node and edge functions, but differing in the number of agents. Analogously, we can define a class of congestion games: the set of congestion games having the same resource functions and the same action choices for every agent, but differing in the number of agents. Finally, since there exists a bijection between congestion games and potential games, we can define a class of potential games as a set of potential games that correspond to all the members of some class of congestion games.

We assume that there are no local effects from unpopulated nodes:

Assumption 4.1 $\forall a \in \mathcal{A}, \forall a' \neq a \in \mathcal{A}$, $\mathcal{F}_{a,a'}(0) = 0$

Also, we make an assumption about the connectivity of G:

Assumption 4.2 $\forall A, B \neq A \in \mathcal{A}$, $(A, B) \in E \Leftrightarrow (B, A) \in E$

That is, each pair of nodes in the graph are either both or neither neighbors of each other, though they might influence each other according to different local-effect functions.

Definition 4.3 *A local-effect game is a* bidirectional local-effect game *(B-LEG) when* $\forall a \in \mathcal{A}, \forall a' \neq a \in \mathcal{A}$, $\mathcal{F}_{a,a'}(x) = \mathcal{F}_{a',a}(x)$.

For B-LEGs local-effect functions between pairs of actions are always the same in both directions: *i.e.*, the local-effect graph G is undirected. Note however that for a given distribution of agents the *magnitude* of the local effects between a pair of actions may be different.

Definition 4.4 *A local-effect game is a* uniform local-effect game *(U-LEG) when* $\forall A, B, C \in \mathcal{A}$ $(B \in neigh(A) \wedge C \in neigh(A)) \rightarrow \forall x\, \mathcal{F}_{A,B}(x) = \mathcal{F}_{A,C}(x)$.

That is, if action A has *any* effect on nodes B and C then the same function governs its effect on both. We define notation for the uniform effect from node A: $\forall y \in \mathcal{A}$, $\mathcal{F}_{A,y} = \mathcal{F}^u_A(x)$.

4.3 Theoretical Results

4.3.1 Nonexistence of Pure Strategy Equilibria

Rosenthal was able to show that congestion games always have a PSNE. For local-effect games, we can find counterexamples where exhaustive enumeration of strategies shows the absence of any PSNE, demonstrating that such a sweeping general result is impossible. One example (found experimentally, and confirmed by exhaustive search) is the B-LEG $\langle\langle\{A, B, C\}, \{(A, B), (B, A), (A, C), (C, A), (B, C), (C, B)\rangle, \mathcal{F}, 11\rangle$, with $\mathcal{F}_{A,A}(x) = 2.79x$, $\mathcal{F}_{B,B}(x) = 4.72x$, $\mathcal{F}_{C,C}(x) = 1.5x$, $\mathcal{F}_{A,B}(x) = 0.64 \log x$, $\mathcal{F}_{A,C}(x) = 0.32 \log x$, $\mathcal{F}_{B,C}(x) = 2.77 \log x$.

4.3.2 Pure Strategy Equilibria: Potential Functions

In this section we show that two interesting classes of local-effect games have potential functions, meaning that they always have pure-strategy Nash equilibria. Although these results show regions of overlap between the class of congestion games and the class of local-effect games, the potential functions themselves are interesting as their construction is nontrivial. Also, these results are useful because they make it possible for the games to be described in the more intuitive local-effect game framework.

Theorem 4.5 *Bidirectional local-effect games have pure strategy Nash equilibria if $\forall i$, $\forall j \neq i$ there exist constants $m_{i,j}$ such that $\mathcal{F}_{i,j}(x) = m_{i,j}x$.*

Proof. Recall that the existence of a potential function entails the existence of a pure-strategy Nash equilibrium. We prove the result by giving a potential function:

$$P(D) = \sum_{i=1}^{n} \sum_{j=1}^{D(i)} \mathcal{F}_{i,i}(j) + \frac{1}{2} \sum_{i=1}^{n} \sum_{j \in neigh(i)} D(i) m_{j,i} D(j) \tag{4.3}$$

The first term is the congestion game potential function for the case where every action names one unique resource. A game with only functions of the form $\mathcal{F}_{i,i}(x)$ is a congestion game, and so must have the congestion game potential function. The relationship between each $\mathcal{F}_{i,j}(x)$ function and the agent's cost function is additive, and potential functions are only used for taking differences. Thus if we can find a potential function P' for a game with only local effects and all $\mathcal{F}_{i,i}(x) = 0$, the potential function for a general B-LEG will be the sum of the congestion game potential function and P'.

Thus it remains to argue that our second term is this P': that it captures changes in utility arising from local effects. Consider the sum of the contribution of local effects to each agent's utility: $s = \sum_{i=1}^{n} \sum_{j \in neigh(i)} D(i) m_{j,i} D(j)$. When a single agent a deviates, s increases by twice the amount of the change to a's utility, because all $\mathcal{F}_{i,j}(x)$ are linear and bidirectional. That is, there is a change both in the amount of local effect acting *on* agent a, and new local effect *caused* by agent a, and bidirectionality and linearity imply that these two amounts are the same. Thus the desired result is obtained by adding $\frac{1}{2}s$ to the congestion game potential function. ■

Observe that Theorem 4.5 holds for B-LEGs with arbitrary node functions $\mathcal{F}_{i,i}(x)$—all that is required is linearity of the local-effect functions.

Theorem 4.6 *Uniform local-effect games have pure strategy Nash equilibria if the local-effect graph is a clique.*

Proof. Again we provide a potential function:

$$P(D) = \sum_{i=1}^{n} \sum_{j=1}^{D(i)} \mathcal{F}_{i,i}(j) - \sum_{i=1}^{n} \sum_{j=1}^{D(i)-1} \mathcal{F}_i^u(j) \tag{4.4}$$

As argued in Theorem 4.5, to construct a potential function it is sufficient to add the standard congestion game potential function with a function that accounts for changes in utility due to local effects. This explains the first term.

Let distributions X and Y be identical except that $D_X(A) = \alpha$ and $D_X(B) = \beta$, while $D_Y(A) = \alpha - 1$ and $D_Y(B) = \beta + 1$. Assuming $\mathcal{F}_{A,A}(\cdot) = \mathcal{F}_{B,B}(\cdot) = 0$, $P(X) - P(Y) = \mathcal{F}_B^u(\beta) - \mathcal{F}_A^u(\alpha - 1)$. This is precisely the change in utility for an agent deviating from A in X to B in Y: the agent will be spared the local effect $\mathcal{F}_B^u(\beta)$ since he moves to B and is no longer subject to its local effect; however, since he moves away from A and the graph is a clique, he is now subject to the local effect

$\mathcal{F}^u_A(\alpha - 1)$. Because the graph is a clique, and because the game is a U-LEG, the argument holds no matter which pair of nodes is chosen as A and B. ■

4.3.3 LEGs and Potential Functions

Finding potential functions is an effective way of proving the existence of pure-strategy equilibria; however, there are many LEGs for which potential functions can be shown not to exist. In this section we give a complete characterization of the class of LEGs which have potential functions. We will state three lemmas that consider arbitrary graphs with every possible subgraph involving three nodes[2], and give sufficient conditions for these LEGs not to have potential functions. Finally, we will combine these lemmas with the positive results in Theorems 4.5 and 4.6 to show that the same conditions are also necessary.

Figure 4.1 shows the graph structure to which Lemma 4.7 refers. There may be any number of nodes other than A, B and C; their connectivity (both with A, B and C and with each other), their node functions and their edge functions are all unrestricted. The only restrictions are that A, B and C must be connected as shown, and that either $\mathcal{F}_{B,C} \neq \mathcal{F}_{C,B}$ or $\mathcal{F}_{B,C}$ is nonlinear.

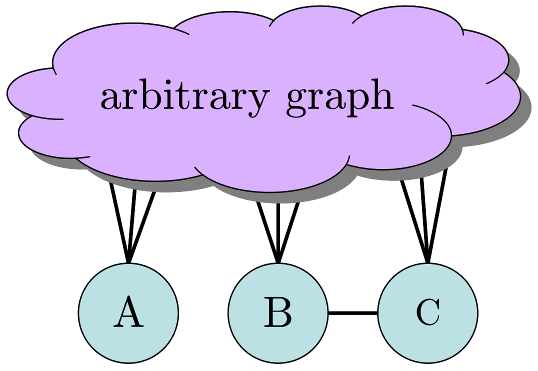


Figure 4.1: Graph structure for Lemma 4.7

[2]Except for completely disconnected graphs, which are trivially congestion games and are handled in Theorem 4.10.

Lemma 4.7 *There exists no class of local-effect games where every game in the class has a potential function, in the case where $\exists A, B, C \in \mathcal{A}$ where $B \in neigh(C)$ and $A \notin neigh(B)$ and $A \notin neigh(C)$ and ($\mathcal{F}_{B,C} \neq \mathcal{F}_{C,B}$ or $\mathcal{F}_{B,C}$ is nonlinear).*

Proof. Assume for contradiction that every LEG in the class[3] has a potential function P. We will consider three distributions of agents in order to derive properties of P. Without loss of generality, we take A, B and C to be the first three actions in the game, and we take the total number of actions to be n. For more compact notation in what follows, let $\alpha = D(A), \beta = D(B)$ and $\gamma = D(C)$. Define the following three distributions: $X = (\alpha, \beta, \gamma, D(4), \ldots, D(n)), Y = (\alpha - 1, \beta + 1, \gamma, D(4), \ldots, D(n))$ and $Z = (\alpha, \beta + 1, \gamma - 1, D(4), \ldots, D(n))$. Without making any assumptions about the local effects between actions A, B and C and any of the other $n - 3$ actions, and for $x \in \{A, B, C\}$, let:

$$U_x(D(x), D(4), \ldots, D(n)) = \mathcal{F}_{x,x}(D(x)) + \sum_{a' \in \{4,\ldots,n\}} \mathcal{F}_{a',a}(D(a')) \tag{4.5}$$

That is, $U_x(D(x), D(4), \ldots, D(n))$ denotes the (negative) utility contributed to each agent taking action $x \in \{A, B, C\}$ by those agents also taking action x, and by those agents taking the 4th through nth actions. For compactness we will abbreviate $U_x(D(x), D(4), \ldots, D(n))$ as $u_x(D(x))$ below.

If distribution X were the case and an agent playing action A switched to action B, then distribution Y would be the result. Thus:

$$P(X) - P(Y) = [u_A(\alpha)] - [\mathcal{F}_{C,B}(\gamma) + u_B(\beta + 1)] \tag{4.6}$$

If X were the case and an agent playing action C switched to action B, then Z would be the result. Thus:

[3]Observe that Lemma 4.7 (like Lemmas 4.8 and 4.9) speak about *classes* of games—*i.e.*, local-effect game structures for which the number of agents is unspecified. This is because there may exist particular numbers of agents for which the contradictions in the lemmas do not obtain for a given graph structure and given node/edge functions. For example, if the game is a B-LEG and all edge functions are linear for integers between 0 and some value x, but nonlinear overall, then potential functions nonetheless exist for all games in the class which have no more than x agents.

$$P(X) - P(Z) = [\mathcal{F}_{B,C}(\beta) + u_C(\gamma)] - [\mathcal{F}_{C,B}(\gamma - 1) + u_B(\beta + 1)] \tag{4.7}$$

If Y were the case and an agent playing action C switched to action A, then Z would be the result. Thus:

$$P(Y) - P(Z) = [\mathcal{F}_{B,C}(\beta + 1) + u_C(\gamma)] - [u_A(\alpha)] \tag{4.8}$$

From equations (4.6) and (4.7), we can infer:

$$\begin{aligned} P(Y) - P(Z) &= [P(X) - P(X)] + P(Y) - P(Z) \\ &= [P(X) - P(Z)] - [P(X) - P(Y)] \\ &= \big[[\mathcal{F}_{B,C}(\beta) + u_C(\gamma)] - [\mathcal{F}_{C,B}(\gamma - 1) + u_B(\beta + 1)]\big] \\ &\quad - \big[[u_A(\alpha)] - [\mathcal{F}_{C,B}(\gamma) + u_B(\beta + 1)]\big] \end{aligned} \tag{4.9}$$

Intersect Equation (4.9) with Equation (4.8) and rearrange. Observe that $u_A(\alpha)$, $u_B(\beta+1)$ and $u_C(\gamma)$ all cancel out, demonstrating that this proof does not depend on what edges exist between A, B, C and the rest of the graph, or on node effects. Define $\mathcal{I}_{a,a'}(x) = \mathcal{F}_{a,a'}(x) - \mathcal{F}_{a,a'}(x - 1)$: the incremental cost on the local effect between a and a' of adding the x^{th} agent to a. We then get:

$$\mathcal{I}_{C,B}(\gamma) - \mathcal{I}_{B,C}(\beta + 1) = 0 \tag{4.10}$$

Clearly, Equation (4.10) will not be satisfied for all β, γ unless $\mathcal{F}_{B,C} = \mathcal{F}_{C,B}$ and $\mathcal{F}_{B,C}$ is linear. This contradicts our assumption that a potential function exists for every LEG in the class. ■

Lemma 4.8 *There exists no class of local-effect games where every game in the class has a potential function, in the case where $\exists A, B, C \in \mathcal{A}$ where $B \in neigh(C)$ and $A \in neigh(B)$ and $A \notin neigh(C)$ and ($\mathcal{F}_{B,C} \neq \mathcal{F}_{C,B}$ or $\mathcal{F}_{B,C}$ is nonlinear or $\mathcal{F}_{A,B} \neq \mathcal{F}_{B,A}$ or $\mathcal{F}_{A,B}$ is nonlinear).*

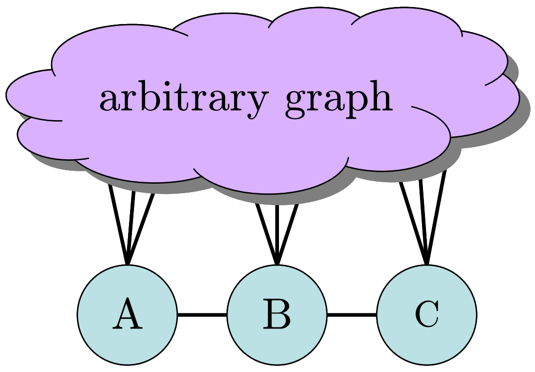


Figure 4.2: Graph structure for Lemma 4.8

Proof. This proof follows the proof of Lemma 4.7 and uses the same setting and definitions, except that (as stated in the theorem and depicted in Figure 4.2) $A \in neigh(B)$.

If distribution X were the case and an agent playing action A switched to action B, then distribution Y would be the result. Thus:

$$P(X) - P(Y) = [\mathcal{F}_{B,A}(\beta) + u_A(\alpha)] - [\mathcal{F}_{A,B}(\alpha - 1) + \mathcal{F}_{C,B}(\gamma) + u_B(\beta + 1)] \quad (4.11)$$

If X were the case and an agent playing action C switched to action B, then Z would be the result. Thus:

$$P(X) - P(Z) = [\mathcal{F}_{B,C}(\beta) + u_C(\gamma)] - [\mathcal{F}_{A,B}(\alpha) + \mathcal{F}_{C,B}(\gamma - 1) + u_B(\beta + 1)] \quad (4.12)$$

If Y were the case and an agent playing action C switched to action A, then Z would be the result. Thus:

$$P(Y) - P(Z) = [\mathcal{F}_{B,C}(\beta + 1) + u_C(\gamma)] - [\mathcal{F}_{B,A}(\beta + 1) + u_A(\alpha)] \quad (4.13)$$

From equations (4.11) and (4.12), we can infer:

$$\begin{aligned}P(Y) - P(Z) = &\big[[\mathcal{F}_{B,C}(\beta) + u_C(\gamma)] - [\mathcal{F}_{A,B}(\alpha) + \mathcal{F}_{C,B}(\gamma - 1) + u_B(\beta + 1)]\big] \\ &- \big[[\mathcal{F}_{B,A}(\beta) + u_A(\alpha)] - [\mathcal{F}_{A,B}(\alpha - 1) + \mathcal{F}_{C,B}(\gamma) + u_B(\beta + 1)]\big] \quad (4.14)\end{aligned}$$

Intersect Equation (4.14) with Equation (4.13) and rearrange:

$$\big[\mathcal{I}_{C,B}(\gamma) - \mathcal{I}_{B,C}(\beta + 1)\big] + \big[\mathcal{I}_{B,A}(\beta + 1) - \mathcal{I}_{A,B}(\alpha)\big] = 0 \quad (4.15)$$

Clearly, Equation (4.15) will not be satisfied for all α, β, γ unless $\mathcal{F}_{A,B} = \mathcal{F}_{B,A}$, $\mathcal{F}_{B,C} = \mathcal{F}_{C,B}$, and both $\mathcal{F}_{A,B}$ and $\mathcal{F}_{B,C}$ are linear. This contradicts our assumption that a potential function exists for every LEG in the class. ∎

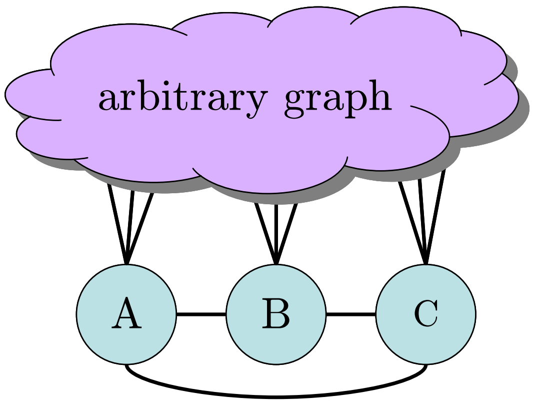


Figure 4.3: Graph structure for Lemma 4.9

Lemma 4.9 *There exists no class of local-effect games where every game in the class has a potential function, in the case where* $B \in neigh(C)$ *and* $A \in neigh(B)$ *and* $A \in neigh(C)$ *and* ($\mathcal{F}_{B,C} \neq \mathcal{F}_{C,B}$ *or* $\mathcal{F}_{B,C}$ *is nonlinear or* $\mathcal{F}_{A,B} \neq \mathcal{F}_{B,A}$ *or* $\mathcal{F}_{A,B}$ *is nonlinear or* $\mathcal{F}_{A,C} \neq \mathcal{F}_{C,A}$ *or* $\mathcal{F}_{A,C}$ *is nonlinear) and* ($\mathcal{F}_{A,B} \neq \mathcal{F}_{A,C}$ *or* $\mathcal{F}_{B,A} \neq \mathcal{F}_{B,C}$ *or* $\mathcal{F}_{C,A} \neq \mathcal{F}_{C,B}$).

Proof. This proof follows the proof of Lemma 4.7 and uses the same setting and definitions, except that (as stated in the theorem and depicted in Figure 4.3) $A \in neigh(B)$ and $A \in neigh(C)$.

If distribution X were the case and an agent playing action A switched to action B, then distribution Y would be the result. Thus:

$$\begin{aligned} P(X) - P(Y) = [&\mathcal{F}_{B,A}(\beta) + \mathcal{F}_{C,A}(\gamma) + u_A(\alpha)] \\ &- [\mathcal{F}_{A,B}(\alpha - 1) + \mathcal{F}_{C,B}(\gamma) + u_B(\beta + 1)] \end{aligned} \tag{4.16}$$

If X were the case and an agent playing action C switched to action B, then Z would be the result. Thus:

$$\begin{aligned} P(X) - P(Z) = [&\mathcal{F}_{A,C}(\alpha) + \mathcal{F}_{B,C}(\beta) + u_C(\gamma)] \\ &- [\mathcal{F}_{A,B}(\alpha) + \mathcal{F}_{C,B}(\gamma - 1) + u_B(\beta + 1)] \end{aligned} \tag{4.17}$$

If Y were the case and an agent playing action C switched to action A, then Z would be the result. Thus:

$$\begin{aligned} P(Y) - P(Z) = [&\mathcal{F}_{A,C}(\alpha - 1) + \mathcal{F}_{B,C}(\beta + 1) + u_C(\gamma)] \\ &- [\mathcal{F}_{B,A}(\beta + 1) + \mathcal{F}_{C,A}(\gamma - 1) + u_A(\alpha)] \end{aligned} \tag{4.18}$$

From equations (4.16) and (4.17), we can infer:

$$\begin{aligned} P(Y) - P(Z) = \big[&[\mathcal{F}_{A,C}(\alpha) + \mathcal{F}_{B,C}(\beta) + u_C(\gamma)] \\ &- [\mathcal{F}_{A,B}(\alpha) + \mathcal{F}_{C,B}(\gamma - 1) + u_B(\beta + 1)]\big] \\ &- \big[[\mathcal{F}_{B,A}(\beta) + \mathcal{F}_{C,A}(\gamma) + u_A(\alpha)] \\ &- [\mathcal{F}_{A,B}(\alpha - 1) + \mathcal{F}_{C,B}(\gamma) + u_B(\beta + 1)]\big] \end{aligned} \tag{4.19}$$

Intersect Equation (4.19) with Equation (4.18) and rearrange:

$$\big[\mathcal{I}_{A,C}(\alpha) - \mathcal{I}_{A,B}(\alpha)\big] + \big[\mathcal{I}_{B,A}(\beta+1) - \mathcal{I}_{B,C}(\beta+1)\big] \\ + \big[\mathcal{I}_{C,B}(\gamma) - \mathcal{I}_{C,A}(\gamma)\big] = 0 \quad (4.20)$$

Equation (4.20) may be rewritten as:

$$\big[\mathcal{I}_{A,C}(\alpha) - \mathcal{I}_{C,A}(\gamma)\big] + \big[\mathcal{I}_{B,A}(\beta+1) - \mathcal{I}_{A,B}(\alpha)\big] + \big[\mathcal{I}_{C,B}(\gamma) - \mathcal{I}_{B,C}(\beta+1)\big] = 0 \quad (4.21)$$

From Equation (4.20) we can see that the contradiction does not obtain for all α, β, γ when $\mathcal{F}_{A,B} = \mathcal{F}_{A,C}, \mathcal{F}_{B,A} = \mathcal{F}_{B,C}$, and $\mathcal{F}_{C,A} = \mathcal{F}_{C,B}$. From Equation (4.21) we can see that the contradiction does not obtain for all $\alpha, beta, \gamma$ when $\mathcal{F}_{A,B} = \mathcal{F}_{B,A}, \mathcal{F}_{B,C} = \mathcal{F}_{C,B}, \mathcal{F}_{A,C} = \mathcal{F}_{C,A}$ and $\mathcal{F}_{A,B}, \mathcal{F}_{A,C}$ and $\mathcal{F}_{B,C}$ are all linear. If neither condition holds, our assumption a potential function exists for every LEG in the class is contradicted. ■

We can now give necessary and sufficient conditions for a class of LEGs to be representable as potential games (and thus, as congestion games).

Theorem 4.10 *There exists a class of potential games that is equivalent to a given class of local-effect games if and only if:*

1. *the local-effect games are bidirectional and all local-effect functions are linear*

2. *the local-effect games are uniform and the local-effect graph is a clique*

Proof. It is easy to show that the stated conditions are sufficient. If the game is bidirectional with linear functions, Theorem 4.5 shows that an equivalent class of potential games exists. Likewise, if the game is uniform and the local-effect graph is a clique, Theorem 4.6 shows that an equivalent class of potential games exists.

Next, we prove that the stated conditions are necessary. We begin by considering classes of LEGs having three or more actions by examining all possible local-effect graph structures. Clearly, we will have considered all possible graphs having three or more nodes if we consider all graphs with no edges, all cliques, and all graphs containing subgraphs having three nodes connected by exactly one or exactly two edges.

If the local-effect graph has no edges, then all LEGs in the class are B-LEGs with linear local-effect functions.

If the local-effect graph contains a three-node subgraph having exactly one edge, and the games are not bidirectional with linear functions, Lemma 4.7 shows that no equivalent class of potential games exists.

If the local-effect graph contains a subgraph with three nodes and exactly two edges, and the games are not bidirectional with linear functions, Lemma 4.8 shows that no equivalent class of potential games exists. If the local-effect graph is a clique, it contains a clique of size three as a subgraph. If this graph is neither uniform nor bidirectional with linear edge functions, Lemma 4.9 shows that no equivalent class of potential games exists.

All that remains is to consider games having exactly one or two actions. A class of games with only a single action is trivially equivalent to a class of potential games because no actions exist to which agents can deviate. All LEGs with only two actions either have no edges, in which case they are B-LEGs with linear local-effect functions, or they have one edge, in which case they are U-LEGs for which the local-effect graph is a clique. ■

4.3.4 Other Pure-Strategy Equilibria

We are able to prove the existence of pure-strategy Nash equilibria for classes of graphs, and node and edge functions that Theorem 4.10 shows cannot have potential functions. The following constructive proof has classes of B-LEGs and U-LEGs as special cases:

Theorem 4.11 *If a local-effect game satisfies:*

1. $\forall A \in \mathcal{A}, \forall B \in neigh(A), \forall x,\ \mathcal{F}_{A,A}(x) \leq \mathcal{F}_{A,B}(x)$
2. $\forall A, B \in \mathcal{A}, \forall x \geq 1,\ \mathcal{F}_{A,B}(x+1) - \mathcal{F}_{A,B}(x) \leq \mathcal{F}_{A,B}(x) - \mathcal{F}_{A,B}(x-1),$

3. $\forall A, B \neq A \in \mathcal{A}, \forall x \geq 1$, $\mathcal{F}_{A,B}(x) - \mathcal{F}_{A,B}(x-1) > 0$,

then there exists a pure-strategy Nash equilibrium in which agents choose nodes that constitute an independent set.[4]

Proof. This proof proceeds by induction, building up a Nash equilibrium one agent at a time, and with each agent making a myopic best response to the previous distribution. In the case of a single agent, it is clearly an equilibrium for him to select the best node. Define D_i as the distribution of agents at induction step i. Assume that $n-1$ agents have each selected the best node in turn, resulting in a distribution D_{n-1} which is a Nash equilibrium and also an independent set. We must show that when an additional agent n chooses the best node the resulting distribution D_n is still an independent set, and still a Nash equilibrium.

First, we show that the new distribution is an independent set. Assume for the purposes of contradiction that it was best for n to choose a node that does not belong to the independent set. Then it must be the case that the selected node has at least one neighbor which has been chosen by one or more other agents. Let the node selected by n be A, and let B be some neighboring node. From condition 2 in the theorem (linearity/sublinearity), we can infer that:

$$F_{B,B}(D_{n-1}(B)+1) \leq F_{B,B}(D_{n-1}(B)) + F_{B,B}(1) \tag{4.22}$$

From condition 1 in the theorem (functional dominance), we know that:

$$F_{B,B}(D_{n-1}(B)) \leq \mathcal{F}_{B,A}(D_{n-1}(B)) \tag{4.23}$$

Thus we can use Equation (4.23) to weaken the bound in Equation (4.22) to get:

$$F_{B,B}(D_{n-1}(B)+1) \leq \mathcal{F}_{B,A}(D_{n-1}(B)) + F_{B,B}(1) \tag{4.24}$$

Define the utility at inductive step i for an agent taking action X, and disregarding any local effect from action Y:

[4]Recall that an independent set is a subset of the nodes in a graph with the property that no two nodes in the subset are neighbors.

$$U^i_{X,\sim Y}(z) = \mathcal{F}_{X,X}(z) + \sum_{W \in neigh(X) | W \neq Y} \mathcal{F}_{W,X}(D_i(W)).$$

At some step i in the induction, $D_i(B) = 0$ and $D_i(A) = 0$, but $D_{i+1}(B) = 1$. From the fact that the distribution of agents resulted from myopic choices (stated in the induction hypothesis), we know that:

$$F_{B,B}(1) \leq U^i_{A,\sim B}(1) \tag{4.25}$$

We can use $U_{A,\sim B}(1)$ in Equation (4.25) because $D_i(B) = 0$ anyway. From condition 3 in the theorem (monotonicity of local-effect functions), and the fact that $i \leq n$ we can write:

$$U^i_{A,\sim B}(1) \leq U^n_{A,\sim B}(1) \tag{4.26}$$

We can use Equation (4.26) to weaken the bound given in Equation (4.25):

$$F_{B,B}(1) \leq U^n_{A,\sim B}(1) \tag{4.27}$$

Finally, we can use Equation (4.27) to further weaken the bound given in Equation (4.24). This gives us:

$$F_{B,B}(D_{n-1}(B) + 1) \leq \mathcal{F}_{B,A}(D_{n-1}(B)) + U^n_{A,\sim B}(1) \tag{4.28}$$

Equation (4.28) contradicts our assumption that agent n would myopically choose A over B; therefore D_n must be an independent set.

Now we show that D' is a Nash equilibrium. Let C be the node that the new agent i selected in making his myopic response to the distribution D. From symmetry of cost functions we know that no agent can profitably deviate from node C: if so, i would have chosen a different node in the first place. Consider an agent j who chose a node $V \neq C$. Agent j's payoff does not change from distribution D to distribution D', because D' is an independent set, and so $\mathcal{F}_{C,V}(\cdot) = 0$ (there are no local effects between nodes C and V. Since distribution D was a Nash equilibrium (inductive

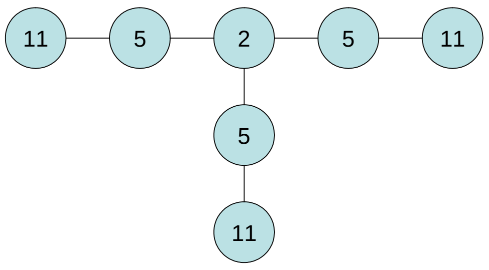


Figure 4.4: T – $k_n = 3$, 50 agents

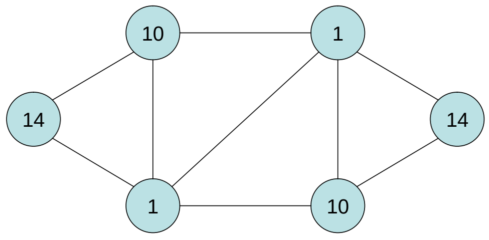


Figure 4.5: Arbitrary – $k_n = 3$, 50 agents

hypothesis) j will not deviate from a new distribution D' that differs only in that node C is more costly. ■

4.4 Empirical Findings

Section 4.3 shows that there are many cases in which local-effect games have pure-strategy Nash equilibria. *Myopic best response* has been shown to be an effective technique for computing pure strategy equilibria in a variety of settings [Monderer & Shapley, 1996]. In this section we show that this simple algorithm can compute pure strategy equilibria for some very large local-effect games that are not covered by any of the positive results in Section 4.3 and that do not have potential functions. We present five different graph structures with similar local-effect functions, and show sample equilibria. We should note that we have been able to find equilibria experimentally for most B-LEGs[5] with different graph structure and different local-effect functions that we have tried, and that convergence occurs within a second in most cases. As with our theoretical results, we do not claim that these equilibria are unique; indeed, because agents' cost functions are symmetric, a new equilibrium can always be constructed from a given equilibrium by swapping action choices between pairs of agents. Furthermore, we have observed many cases where multiple structurally different equilibria exist in the same local-effect game.

[5]So far, we have only experimented with B-LEGs because undirected local-effect graphs are easier to specify and generate, and because we consider them to be among the most natural LEGs. We expect to experiment with other classes of LEGs in our future work.

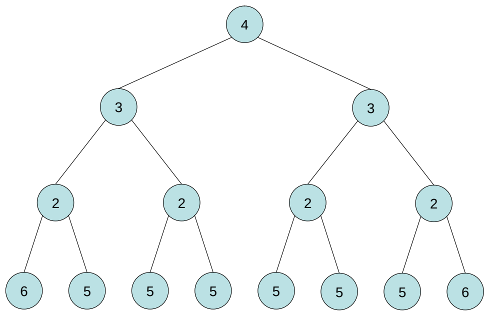


Figure 4.6: Binary Tree – $k_n = 6$, 60 agents

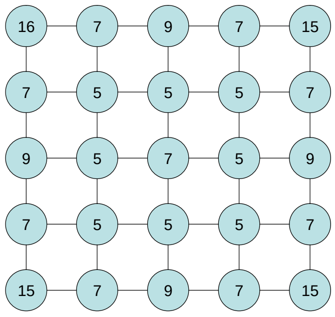


Figure 4.7: Grid – $k_n = 4$, 200 agents

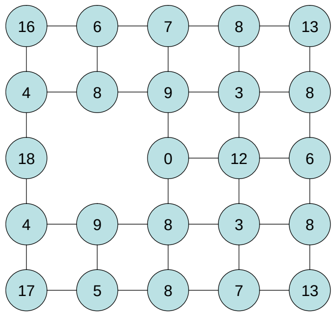
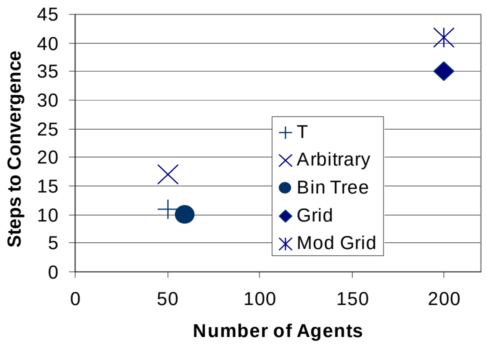


Figure 4.8: Modified Grid – $k_n = 4$, 200 agents

All games shown here are B-LEGs with $\forall A, \forall B\ \mathcal{F}_{A,B}(x) = k_{A,B} \log(x+1)$. We use one k_n for all node functions and another k_e for all edge functions (*i.e.*, $\forall A, \forall B \neq A\ k_{A,A} = k_{B,B}$ and $\forall A, \forall B \neq A, \forall C \notin \{A,B\}\ k_{A,B} = k_{A,C}$). We hold $k_e = 1$ throughout, and vary k_n to highlight some of the more interesting equilibria. These equilibria are representative of average runs, and were found with a minimum of parameter manipulation. Each node is labelled with the number of agents choosing the node in equilibrium. Figure 4.4 shows a T structure representative of a simple location problem. Figure 4.5, which we call 'arbitrary' in what follows, is interesting because there are 2 nodes with 2 neighbors, 2 with 3 neighbors and 2 with 4 neighbors. This setting could represent a role formation game. Figure 4.6 shows a binary tree structure; observe that most agents select leaf nodes because they have



.25in

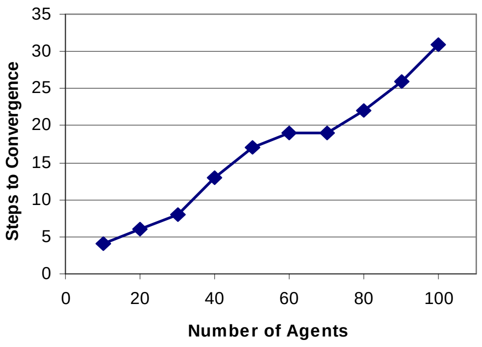


Figure 4.9: Steps to convergence for the five graphs

Figure 4.10: Steps to convergence for the arbitrary graph

only one neighbor, and thus the parents of leaves are chosen by few agents. Figure 4.7 shows a two-dimensional grid, representative of our coffee house location problem. Observe that the corners are most desirable, as they have only two neighbors; nodes neighboring corners are thus under-populated, leading to another concentration of agents in the middle of each edge. It is also interesting that agents concentrate in the central node, even though it has four neighbors, because its neighbors are relatively under-populated. Figure 4.8 shows what happens to the game from Figure 4.7 when we remove a single node (consider the same location problem when one node becomes unavailable). Observe that agents generally cluster around the missing node, except for one neighboring node that is entirely unpopulated, as a result of the large local effects acting upon it.

The amount of time it took to reach convergence in each graph is shown in Figure 4.4, starting in each case with a uniform distribution of agents across the actions. Finally, since the 'arbitrary' graph in Figure 4.5 took the longest to converge (34% of the agents moved before convergence occurred) we examine this graph in more detail in Figure 4.4. Observe that as we vary the number of agents, the number of steps required for convergence increases roughly linearly.

4.5 Conclusions

To encourage future work and to summarize the local-effect game setting, we restate the most important conditions that must be satisfied in order for a game to be representable as a LEG:

1. **Symmetry**: every agent has the same action choices and utility function;
2. **Anonymity**: every agent affects other agents in the same way; thus, we can ignore agent identities and consider only the number of agents taking each action;
3. **Additivity**: the relationship between different local effects acting on the same node is linear;
4. **Connected Neighbors**: if A is connected to B in the local-effect graph then B is also connected to A. However, each edge can potentially be labelled with a different function.

Local-effect games exploit context-specific independence between players' payoff functions. Finding a potential function is a good technique for finding equilibria; we identify all the local-effect games for which potential functions exist, and provide the potential function in each case. We also give a positive theoretical result for a broad class of games that do not have potential functions. Furthermore, we can show that myopic best response dynamics converge quickly in some other cases in which potential functions do not exist, and which are not covered by our positive theoretical results.

Part II

Computational Issues in Multiagent Resource Allocation

Chapter 5

Combinatorial Auctions

5.1 Motivation

Auctions are the most widely studied mechanism in the mechanism design literature in economics and game theory [Fudenberg & Tirole, 1991]. This is due to the fact that auctions are basic protocols, serving as the building blocks of more elaborated mechanisms. Given the wide popularity of auctions on the Internet and the emergence of electronic commerce, where auctions serve as the most popular game-theoretic mechanism, efficient auction design has become a subject of considerable importance for researchers in multi-agent systems (*e.g.*, [Wellman *et al.*, 1998; Monderer & Tennenholtz, 2000]). The use of auctions in business-to-business trades is also increasing rapidly. Within AI there is growing interest in using auction mechanisms to solve resource allocation problems in competitive multiagent systems. For example, auctions and other market mechanisms are used in network bandwidth allocation, distributed configuration design, factory scheduling, and operating system memory allocation [S. H. Clearwater, 1996; Wellman, 1993].

5.1.1 Complementarity

The value of a good to a potential buyer can depend on what other goods s/he wins. We say that there exists *complementarity* between goods γ_1 and γ_2 to agent a if

$u_a(\{\gamma_1, \gamma_2\}) > u_a(\{\gamma_2\}) + u_a(\{\gamma_2\})$, where $u_a(S)$ is the utility to a of acquiring the set of goods S. If goods γ_1 and γ_2 were auctioned separately, it is likely that neither of the typically desired properties for auctions—efficiency or revenue maximization—would hold. One way to accommodate complementarity in auctions is to allow bids for combinations of goods as well as for individual goods. *Combinatorial auctions* are a class of auctions that accommodate bidders whose valuations exhibit complementarities: multiple goods are auctioned simultaneously and bidders place as many bids as they want for different bundles of goods with the guarantee that these bundles will be allocated "all-or-nothing". For example, imagine an auction of used electronic equipment. A bidder might value a particular TV at x and a particular VCR at y, but value the pair at $z > x + y$.

5.1.2 Substitutability

It is also common for bidders to desire a second good less once they have won a first good. We say that there exists *substitutability* between goods γ_1 and γ_2 to agent a when $u_a(\{\gamma_1, \gamma_2\}) < u_a(\{\gamma_1\}) + u_a(\{\gamma_2\})$. A common example of substitutability is for a bidder to be indifferent between several goods but not to want more than one. In order to be useful, a combinatorial auction mechanism should provide some way for bidders to indicate that goods (or, more generally, bundles of goods) are substitutable. Combinatorial auctions that allow the description of valuation functions involving both complementarity and substitutability can be seen as providing a general framework for allocation and decision-making problems among agents in competitive multiagent systems.

5.1.3 Applications

Combinatorial auctions are applicable to many real-world situations. In an auction for the right to use railroad segments a bidder desires a bundle of segments that connect two particular points; at the same time, there may be alternate paths between these points and the bidder needs only one [Brewer & Plott, 1996]. Similarly, in the FCC spectrum auction bidders may desire licenses for multiple geographical regions at the same frequency band while being indifferent to which particular band they receive [Milgrom, 1998]. These and other real-world applications of combinatorial auctions are described in more detail in Chapter 7. While economics and game theory provide many insights into the potential uses of such auctions, they have little to say about computational considerations. This is an obstacle to the practical use of combinatorial auctions, because it turns out that combinatorial auctions often give rise to very hard computational problems.

5.2 Combinatorial Auction Winner Determination

In a combinatorial auction, a seller is faced with a set of price offers for various bundles of goods, and his aim is to allocate the goods in a way that maximizes his revenue. The *winner determination problem* (WDP) is choosing the subset of bids that maximizes the seller's revenue, subject to the constraint that each good can be allocated at most once.

5.2.1 Formal definition

Let $G = \{\gamma_1, \gamma_2, \dots, \gamma_m\}$ be a set of goods, and let $B = \{b_1, \dots, b_n\}$ be a set of bids. Bid b_i is a pair $(p(b_i), g(b_i))$ where $p(b_i) \in \mathbb{R}^+$ is the price offer of bid b_i and $g(b_i) \subseteq G$ is the set of goods requested by b_i. For each bid b_i define an indicator variable x_i that encodes the inclusion or exclusion of bid b_i from the allocation.

Problem 5.1 *The* single-unit WDP *is the following integer program:*

$$\begin{aligned}
\text{maximize:} \quad & \sum_i x_i p(b_i) & \\
\text{subject to:} \quad & \sum_{i|\gamma \in g(b_i)} x_i \leq 1 & \forall \gamma \in G \\
& x_i \in \{0, 1\} & \forall i
\end{aligned}$$

5.2.2 XOR constraints

Problem 5.1 allows bidders to describe complementarities in their valuations; however, it does not explicitly allow the expression of substitutabilities. To do so, bidders must have some way of indicating that their interest in two bundles is mutually exclusive. Let $S = \{s_1, \ldots, s_\kappa\}$, where each s_i denotes a set of bids which are not allowed to be allocated together by the optimization algorithm.

Problem 5.2 *The* single-unit WDP with XOR constraints *is the following integer program:*

$$\begin{array}{lll} \text{maximize:} & \sum_i x_i p(b_i) & \\ \text{subject to:} & \sum_{i \mid \gamma \in g(b_i)} x_i \leq 1 & \forall \gamma \in G \\ & \sum_{i \mid b_i \in s_j} x_i \leq 1 & \forall j \in [1, \kappa] \\ & x_i \in \{0, 1\} & \forall i \end{array}$$

Some algorithms may not provide a way of specifying sets of mutually exclusive bundles s. Luckily, it is possible to use an encoding trick, introducing *dummy goods* [Fujishima *et al.*, 1999]. Dummy goods do not correspond to actual goods in the auction, but serve to enforce mutual exclusion between bids. For example, if bids b_1 and b_2 are intended to be mutually exclusive, we add a dummy good d to each bundle, defining new bids b'_i where $g(b'_i) = g(b_i) \cup d$. Since the good d can be allocated only once, at most one of b'_1 and b'_2 will allocated. More generally, it is possible to introduce n-unit dummy goods to enforce the condition that no more than n of a set of bids may be allocated (see Section 5.2.3 for a definition of units), and to use multiple dummy goods with the same bundles to enforce other complex constraints. This technique can lead to a combinatorial explosion in the number of bids if many goods are substitutable, but in many interesting cases this does not occur. While dummy goods increase the expressive power of the bidding language, making use of them requires no changes to an optimization algorithm. In fact, observe that Problem 5.2 is exactly the same as Problem 5.1 when XOR constraints are expressed using dummy goods. Hence, in what follows we do not discuss explicit XOR constraints, but assume that dummy goods are used where required.

5.2.3 Multi-unit auctions

In some cases, the set of goods at auction will contain subsets of goods among which all bidders are indifferent. We call these subsets *units* of a single good. While it is possible to auction each unit as a separate good, this forces bidders interested in a subset of the units to specify unnecessary XOR bids. For example, consider an electronics manufacturer auctioning 100 identical TVs and 100 identical VCRs. A retailer who wants to buy 70 TVs and 30 VCRs would be indifferent between all bundles having 70 TVs and 30 VCRs. Rather than having to bid on each of the $\binom{100}{70} \cdot \binom{100}{30}$ distinct bundles, she would prefer to place the single bid (price, {70 TVs, 30 VCRs}). This can be achieved by generalizing Problem 5.1. Let $q(\gamma)$ denote the number of units of good γ.

Problem 5.3 *The* multi-unit WDP *is the following integer program:*

$$\begin{array}{lll} \text{maximize:} & \sum_i x_i p(b_i) & \\ \text{subject to:} & \sum_{i|\gamma \in g(b_i)} x_i \leq q(\gamma) & \forall \gamma \in G \\ & x_i \in \{0,1\} & \forall i \end{array}$$

5.2.4 Asymptotic Hardness

With or without XOR constraints, the WDP is equivalent to weighted set-packing and is therefore $\mathcal{NP}$-hard even in its single-unit variant: see *e.g.*, [Rothkopf *et al.*, 1998]. Furthermore, it is known that the WDP is inapproximable within any constant factor: see *e.g.*, [Sandholm, 1999].

5.3 Related Work on the WDP

In recent years many researchers have been interested in the combinatorial auction winner determination problem. For a survey, see [de Vries & Vohra, 2003].

5.3.1 Tractable Subcases

Rothkopf *et al.* [1998] identified the following subcases of the single-unit WDP which may be solved in polynomial time:

1. bids contain no more than two goods;
2. for any two bids, either they are disjoint or one is a subset of the other; or
3. bids only name goods that are consecutive given a one-dimensional ordering.[1]

More recent work on tractable subcases may be found in [Nisan, 2000; Tennenholtz, 2000; de Vries & Vohra, 2003]. The case of infinitely divisible goods may be solved in polynomial time by using linear programming techniques: we solve the linear programming relaxation of Problem 5.1, where the integrality constraint is replaced by the linear constraint $0 \leq x_i \leq 1$.

5.3.2 Approximation algorithms

Some researchers have studied algorithms to approximate the WDP, despite the fact that the WDP cannot be approximated with guarantees. For example, Hoos and Boutilier [2000] and Zurel and Nisan [2000] have proposed algorithms with good empirical performance despite their lack of theoretical guarantees. Furthermore others,

[1]In fact, this case can be extended to the case where goods are placed around a ring and each bid requests only consecutive goods. Consider adding each bid in turn (and removing all other bids that conflict with this bid): the remaining subproblem to be solved has only bids that request consecutive goods given a one-dimensional ordering, because the selection of the first bid breaks the ring.

notably Nisan and Ronen [2000] and Lehmann *et al.* [1999], have proposed alternative economic mechanisms that are built around approximation algorithms. It is also possible to make bidders responsible for improving the quality of the approximation. Banks *et al.* [1989] and Bykowsky *et al.* [1995] have reported a mechanism called AUSM in which non-winning bids are pooled in a stand-by queue. Bidders can combine their bids with other bids currently in the queue to form new allocations. A new allocation is adopted if it generates more revenue than the previously best allocation.

5.3.3 Solving the WDP to optimality

Although the WDP is $\mathcal{NP}$-hard, in practice it is possible to address interestingly-large datasets with heuristic methods [Fujishima *et al.*, 1999; Sandholm, 1999; Gonen & Lehmann, 2000; Leyton-Brown *et al.*, 2000b; Nisan, 2000; Sandholm & Suri, 2000; Gonen & Lehmann, 2001; Sandholm *et al.*, 2001]. Furthermore, there are reasons why it can be important to solve the WDP to optimality, and why restrictions to a tractable subcase may not be acceptable. For example, optimal solutions to the WDP are required in order for the Vickrey-Clarke-Groves mechanism [Mas-Colell *et al.*, 1995; Varian, 1995] give rise to dominant strategies. In fact, Nisan and Ronen [2000] show that the Vickrey-Clarke-Groves mechanism can give rise to arbitrarily bad outcomes when agents suspect that there is any possibility that any non-optimal solution to the WDP will be used. As another example, Parkes [1999], among others, has proposed an ascending auction mechanism that requires a provably-optimal solution to the WDP. Because of the importance of such applications, we concern ourselves in this work only with provably-optimal solutions to the WDP.

Chapter 6

Algorithms for Solving the Combinatorial Auction Winner Determination Problem

We present two branch-and-bound algorithms that exploit instances' particular bid structures to find optimal solutions to the WDP, using contextual information to make upper bounds tighter. Upper bounds are further tightened by online caching of results from unpruned subtrees. The first algorithm, CASS, considers only the single-unit WDP (*i.e.*, it solves WDP Problem 5.1). The second algorithm, CAMUS, generalizes CASS, addressing the multi-unit WDP (Problem 5.3).

6.1 CASS Algorithm

In this section we present an algorithm, Combinatorial Auction Structured Search (CASS), a branch-and-bound search algorithm with a novel heuristic. Most importantly, CASS structures the search space in a way that provides context to this heuristic in order to allow more pruning during the search and that avoids consideration of most infeasible allocations. CASS also caches the results of partial searches and prunes the search tree. Finally, CASS may be used as an anytime algorithm, as it tends to find good allocations quickly.

Before proceeding, we must introduce additional notation pertaining to allocations. An allocation $\pi \subseteq B$ is a subset of the bids where $\forall b_1 \in B, b_2 \neq b_1 \in B$, $g(b_1) \cap g(b_2) = \{\}$. We overload the functions $g()$ and $p()$ to apply to allocations: $g(\pi) = \bigcup_{b \in \pi} g(b)$ and $p(\pi) = \sum_{bin\pi} p(b)$. A full allocation π_{full} is an allocation for which $g(\pi_{full}) = G$, and a partial allocation is an allocation that is not full.

6.1.1 Dominated Bids

Some bids may be removed in a polynomial-time preprocessing step before search begins. For each pair of bids (b_1,b_2) where $g(b_1) \subseteq g(b_2)$ and $p(b_1) \geq p(b_2)$, we may remove b_2 from the list of bids to be considered during the search as b_2 is never preferable to b_1 (hence we say that b_1 *dominates* b_2).

6.1.2 Branch-and-Bound Search

Branch-and-bound search is a general search strategy that is widely used in the operations research community (see *e.g.*, [Nemhauser & Wolsey, 1988]). We explain it here using the terminology of combinatorial auctions.

Whenever a bid is encountered that does not conflict with the current partial allocation then the search tree branches, where one branch adds the bid to the partial allocation and the other does not. CASS performs a depth-first search, meaning that one branch of the tree is fully explored before the other is considered. (This has the advantage that CASS requires only linear space to store the search tree.) When a full allocation π is reached CASS records this allocation as π_{best} if $p(\pi) > p(\pi_{best})$, and then backtracks.

CASS also computes a function $h(\pi)$ at each node, which gives an upper bound on the revenue that can be collected from the goods $G \setminus g(\pi)$. When $h()$ indicates that the current subtree cannot lead to a solution better than π_{best} then the search tree can be *pruned*: we can backtrack before a full allocation has been constructed. More precisely, we backtrack whenever $p(\pi) + h(\pi) \leq p(\pi_{best})$. We will describe the construction of the function $h(\pi)$ in Section 6.1.4, but first we must introduce the concept of bins.

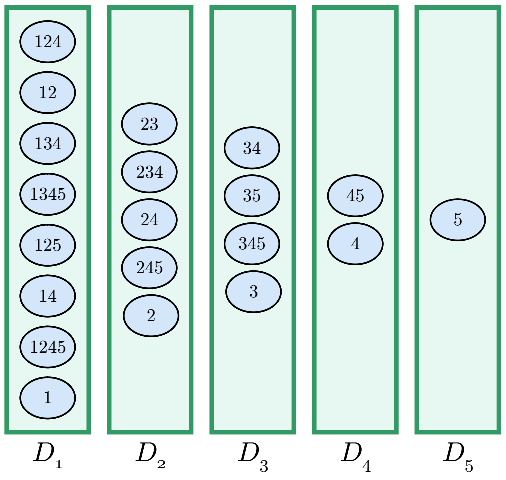


Figure 6.1: Partition into Bins

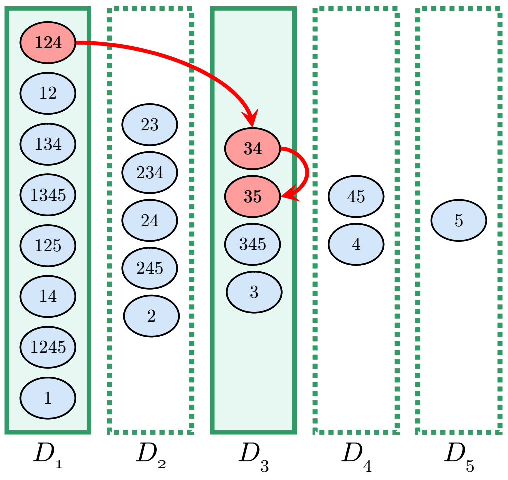


Figure 6.2: Skipping bins

6.1.3 Bins

We can reduce the number of infeasible allocations considered by partitioning bids into *bins*. First, we must choose an ordering on the goods. We create one bin for each good, and we place each bid into the bin corresponding to its lowest-order good. For an example, see Figure 6.1. The example shows input from an auction with five goods, $G = \{1, 2, 3, 4, 5\}$. Circles in the figure represent bids, the concatenated numbers in the circles represent the goods named in each bid, and prices are omitted from the bids for clarity.

Rather than always trying to add each bid to our allocation, we add at most one bid from every bin since all bids in a given bin are mutually exclusive. In fact, we can often skip bins entirely. While considering bin D_i, if we observe that good $j > i$ is already part of the allocation then we do not need to consider any of the bids in D_j. In general, instead of considering each bin in turn, skip to D_k where $k \notin g(\pi)$ and $\forall i < k$, $i \in g(\pi)$. To avoid the situation where it is possible to enter a bin that we cannot leave, we augment the set of bids. If there is no bid requesting a single unit of any good $\gamma_j \in G$ then we add a dummy bid $b = (0, \gamma_j)$.

Continuing our example, see Figure 6.2. The first bid we add contains goods $\{1, 2, 4\}$, so we skip all remaining bids in bin D_1 and all bids in bin D_2, since all of these bids will conflict. We thus consider bin D_3. The first bid we encounter requests goods $\{3, 4\}$, which also conflicts with our partial allocation. We move to the next bid in D_3, which requests goods $\{3, 5\}$. We have found our first full allocation, so we update our lower bound and backtrack.

The main benefit of bins is not the ability to avoid consideration of conflicting bids, however. Bins are powerful because they allow the pruning function to consider context without significant computational cost. If bids in bin D_i are currently being considered then the pruning function must only take into account bids in bins $\{D_i, \ldots, D_m\}$. Since most bids will belong to a low-order bin[1] but the search will spend most of its time in higher-order bins, this can allow us to generate very tight upper bounds. Furthermore, because the partitioning of bids into bins is fixed we can compute the upper bound information for each bin in a preprocessing step; this makes the upper bound fast to evaluate during search.

6.1.4 Upper Bound

We now describe the construction of the function $h(\pi)$. For every remaining good j we calculate a value $v(j)$, the maximum over all the remaining bids requesting good j that do not conflict with π of the bid's price divided by the number of goods requested by the bid. The sum of $v(j)$ values for all goods is an upper bound on optimal revenue because it relaxes the constraint that the bids in the optimal allocation may not conflict.

More formally, consider that we have built up a partial allocation π and reached bin D_σ. For each bid b_i, let $a(b_i) = p(b_i)/m$ be the average price per good of bid b_i. Notice that the average price per good may change dramatically from bid to bid, and it is a non-trivial notion; our technique will work for any arbitrary average price per good. Let $L_\sigma(j)$ be a list of the bids that refer to good j and that belong to bins D_ϕ with $\phi \geq \sigma$ (*i.e.*, the bids that can be encountered in the remainder of the search). The list is sorted in a monotonically decreasing manner according to the a_i's. Observe

[1]To see why bids are not spread evenly across the bins, consider what happens when we receive a bid for *every* bundle. Half of the bids will involve good γ_1 and will thus belong to bin D_1; half the bids that do not involve γ_1 will involve γ_2 and belong to bin D_2, and so on. Clearly, each bin D_{i+1} will contain half as many bids as its predecessor D_i.

that $L_\sigma(j)$ can be precomputed before search begins. $v(j)$ is defined as $a(b_j^{\text{ok}})$, where b_j^{ok} is the first bid in $L_\sigma(j)$ that does not conflict with π.

Theorem 6.1 *Let* $B^* = \{b_1^*, \ldots, b_s^*\}$ *be the bids in an optimal allocation. Then,* $r^* = \Sigma_{b \in B^*} p(b) \leq \Sigma_{1 \leq j \leq m} v(j)$.

Remark: In the multi-unit special case where when all goods happen to have only one unit, the upper bound function in Section 6.2.4 computes exactly the same upper bounds as the function presented above. Thus Theorem 6.2 considers a more general case, and so the proof to Theorem 6.1 can be deduced from that proof.

6.1.5 Caching

CASS's caching scheme is a form of dynamic programming that allows the algorithm to use experience from earlier in the search to tighten its upper bound function; it is illustrated in Figure 6.3. Consider a partial allocation π_1 that is reached during the search phase. If the search proceeds beyond π_1 then $h(\pi_1)$ was not sufficiently small to allow us to backtrack. Later in the search we may reach an allocation π_2 which, by combining different bids, achieves the same allocation π_1. CASS incorporates a mechanism for caching the results of the search beyond π_1 to generate a better estimate for the revenue given π_2 than is given by $h(\pi_2)$. (Since π_1 and π_2 do not differ in the units of goods that remain, $h(\pi_1) = h(\pi_2)$.) Consider all the allocations extending π_1 upon consideration of which the algorithm backtracked, denoted $s_1, \ldots, s_f$. When we backtracked at each s_i we did so because $p(s_i) + h(s_i) \leq p(\pi_{best})$, as explained above, or because s_i was a full allocation. It follows that $max_i(p(s_i) + h(s_i)) - p(\pi_1)$ is an overestimate of the revenue attainable beyond π_1, and that it is a smaller overestimate than $h(\pi_1)$. If it were not, we would have backtracked at π_1 rather than searching this subtree. We cache the value $c(g(\pi)) = max_i(p(s_i) + h(s_i)) - p(\pi_1)$ and backtrack when $p(\pi_2) + c(g(\pi_2)) \leq p(\pi_{best})$.

Our cache is implemented as a hash table, since caching is only beneficial to the overall search if lookup time is inconsequential. A consequence of this choice of data structure is that cache data may sometimes be overwritten; we overwrite an old entry

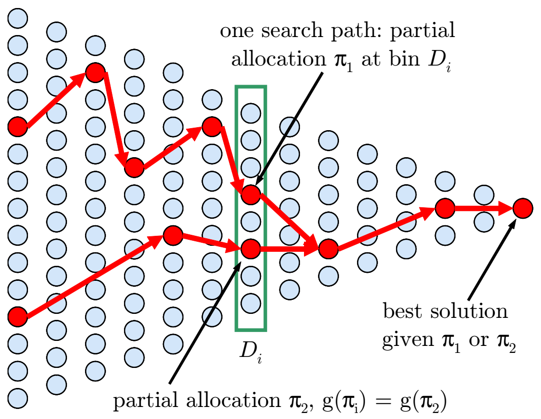

one search path: partial allocation π_1 at bin D_i
best solution given π_1 or π_2
D_i
partial allocation π_2, $g(\pi_1) = g(\pi_2)$

Figure 6.3: Caching

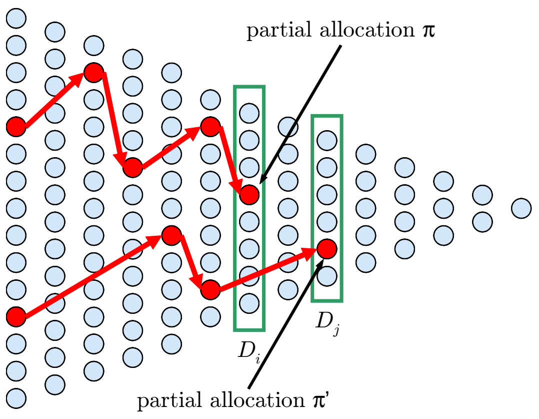


Figure 6.4: Cache Pruning

in the cache when the search associated with the new entry examined more nodes.[2] Even when we do overwrite useful data the error is not catastrophic, however: in the worst case we must simply search a subtree that we might otherwise have pruned. In order to reduce the cost of writing useless entries to the cache, and to reduce the chance that useful entries will be overwritten, partial allocations are only stored in the cache if the search tree below that point involved at least one backtrack. (Because of the exponential character of search trees, the vast majority of nodes that are eligible for caching will be one step away from leaf nodes, which means that caching these nodes would give a negligible performance gain.)

We can also use the cache to prune even when we reach a new partial allocation π_2 that has never been reached before, as shown in Figure 6.4. The search path is provably unable to lead to a new best allocation whenever $g(\pi_1) \subset g(\pi_2)$ and $p(\pi_2)+c(g(\pi_1)) \leq p(\pi_{best})$. (Since our cache is implemented as a hash table, we detect this case by checking the cache for each π_1 that differs from π_2 by the exclusion of each a single good; this requires a linear number of cache lookups.) In this case, the sum of the revenue from the cached path beyond π_1 and the revenue leading up to π_2 is less than the revenue from π_{best}. Note that since π_2 allocates a superset of the goods allocated in π_1, the goods in $g(\pi_2) \setminus g(\pi_1) \cap g(\pi_2)$ are counted both in $p(\pi_2)$

[2]We must, however, store $g(\pi)$ in the hash table along with $c(g(\pi))$, so that we can detect cache collisions.

and $c(g(\pi_1))$, corresponding to an overestimate of revenue.[3] Therefore, no allocation better than π_{best} could be found by expanding the search tree below π_2.

6.1.6 Good Ordering Heuristic

We must determine an ordering of the goods; that is, decide which good will correspond to the first bin, which will correspond to the second, etc. For each good i we compute $score_i = numbids_i/avggoods_i$, where $numbids_i$ is the number of bids that request good i and $avggoods_i$ is the average number of goods requested by these bids. We designate the lowest-order good as the good with the lowest score, then we recalculate the score for the remaining goods and repeat. The intuition behind this heuristic is as follows:

- We want to minimize the number of bids in low-order bins, to minimize early branching and thus to make each individual prune more effective.
- We want to maximize the total number of goods requested by bids in low-order bins. Taking these bids moves us more quickly towards the leaves of the search tree, again providing the pruning function with more contextual information.

6.1.7 Bid Ordering Heuristic

Our second heuristic determines the ordering of bids within bins. We sort bids dynamically—*i.e.*, bids may be ordered differently when there is a different partial allocation π. Bids are sorted in a given bin in descending order of $score(b_j)$, where (abusing notation slightly):

$$score(b_j) = \frac{p(b_j)}{|g(b_j)|} + h(\pi \cup b_j).$$

The intuition behind this heuristic is that the average price per good of b_j is a measure of how promising the bid is, while the upper bound $h(\pi \cup b_j)$ is an estimate of

[3]In fact, this bound can be tightened. Let $\mathcal{G} = g(\pi_2) \setminus g(\pi_1) \cap g(\pi_2)$. We can actually backtrack whenever $p(\pi_2) + c(g(\pi_1)) - \ell(\pi_2, \mathcal{G}) \leq p(\pi_{best})$, where $\ell(\pi_2, \mathcal{G})$ is a lower bound on the revenue that could be achieved from the goods $\mathcal{G}$ given the allocation π. A simple implementation of $\ell()$ would be the sum of the singleton bids for each of the goods in $\mathcal{G}$.

```
Process dominated bids.
Determine an ordering on the goods, according to the good-ordering
 heuristic.
Partition all bids into bins, according to the good ordering.
Precompute pruning information for each bin.
Set $i = 1$ and $\pi = \{\}$.
Recursive entry point:
    $\pi = \pi \cup b_j$.
    If ($p(\pi) + c(g(\pi)) \leq p(\pi_{best})$) backtrack.
    If ($p(\pi) + h(\pi) \leq p(\pi_{best})$) backtrack.
    If ($|goods(\pi)| = total$) update $\pi_{best}$ if necessary; backtrack.
    Set $i$ to the index of the lowest-order good absent from $\pi$.
    Dynamically order the bids in bin $i$, and remove bids that
     conflict with $\pi$.
    Recurse to the recursive entry point above.
    $\pi = \pi \setminus b_j$.
Return the optimal allocation: $\pi_{best}$.
```

Figure 6.5: CASS Pseudocode

how promising the unallocated units are, given the partial allocation. This heuristic helps CASS to find good allocations quickly, improving anytime performance and also increasing π_{best}, making pruning more effective. Observe that dynamic reordering of the goods in each bin allows us to make use of an upper bound which depends on π.

6.2 CAMUS Algorithm

We now present a generalization of CASS which can solve Problem 5.3: the general multi-unit WDP. This algorithm is termed CAMUS (Combinatorial Auction Multi-Unit Search), and was introduced in [Leyton-Brown *et al.*, 2000b]. We explain characteristics of CAMUS that differ from CASS, and then give pseudocode for CAMUS.

In this section we make use of the following notation specific to the multi-unit case. Let $units(j)$ denote the total number of units of good j. Redefine the set of bids $B = \{b_1, \ldots, b_n\}$ so that bid b_i is a pair $(p(b_i), e(b_i))$, where $e(b_i) = (e(b_i)_1, e(b_i)_2, \ldots, e(b_i)_m)$ and $e(b_i)_j$ is the number of requested units of good γ_j in b_i. We overload the function *units* to refer to allocations: given an allocation π we denote the total number of units allocated as $units(\pi)$, and given both an allocation π and a good γ_i we denote the total number of units of good γ_i allocated in π by $units_i(\pi)$.

6.2.1 Dominated Bids

In the multi-unit case we must handle domination in a different way than we did in the single-unit case. We say that bid b_1 dominates b_2 whenever $p(b_1) \geq p(b_2)$ and $e(b_1)_j \leq e(b_2)_j$ for every good j. Although b_2 is never preferable to b_1, it is possible that an optimal allocation could contain both b_1 *and* b_2. For this reason we store b_2 in a secondary data structure associated with b_1, and consider adding it to those allocations which include b_1.

6.2.2 Subbins

In the multi-unit setting, we will often need to select more than one bid from a given bin. This leads to the idea of *subbins*. A subbin is a subset of the bids in a bin that is constructed during the search. Since subbins are created dynamically they cannot provide precomputed contextual information; rather, they facilitate the efficient selection of multiple bids from a given bin. Every time we add a bid to our partial allocation we create a new subbin containing the next set of bids to consider. If the search moves to a new bin, the new subbin is generated from the new bin by removing all bids that conflict with the current partial allocation. If the search remains in the same bin, the new subbin is created from the current subbin by removing conflicting bids as above, and additionally: if $b_1, b_2, \ldots, b_i$ is the ordered set of elements in the current subbin and b_j is the bid that was just chosen, then we remove all $b_k, k \leq j$. In this way we consider all combinations of non-conflicting bids in each bin rather than all permutations.

6.2.3 Dynamic Programming

Singleton bids (that is, bids that name units from only one good) deserve special attention. These bids will generally be among the most computationally expensive to consider—the number of nodes to search after adding a very short bid is nearly the same as the number of nodes to search after skipping the bid, because a short bid allocates few units and hence conflicts with few other bids. Unfortunately, we expect that singleton bids will be quite common in a variety of real-world multi-unit CA's. CAMUS simplifies the problem of singleton bids by applying a polynomial-time dynamic programming technique as a preprocessing step. We construct a vector $singleton_\gamma$ for each good γ, where each element of the vector is a set of singleton bids naming only good γ. $singleton_\gamma(j)$ evaluates to the revenue-maximizing set of singleton bids totaling j units of good γ. This frees us from having to consider singleton bids individually; instead, we consider only elements of the singleton vector and treat these elements as atomic bids during the search. Also, there is never a need to add more than one element from each singleton vector. To see why, imagine that we add both $singleton_\gamma(j)$ and $singleton_\gamma(k)$ to our partial allocation. These two elements may have bids in common, and additionally there may be singleton bids with more than $max(j, k)$ elements that would not conflict with our partial allocation but that we have not considered. Clearly, we would be better off adding the single element $singleton_\gamma(j + k)$.

We now show how to construct the *singleton* vector. Let $b_1, \ldots, b_\ell$ be bids for a single good γ. Our aim is to compute the optimal selection of b_i's in order to allocate k units of good γ, for $1 \leq k \leq units(\gamma)$. Consider a two dimensional grid of size $[1 \ldots \ell] \times [1 \ldots units(\gamma)]$ where the (i, j)-th entry, denoted by $U(i, j)$, is the optimal allocation of j units considering only bids $b_1, \ldots, b_i$. The value of $U(i, j)$, denoted by $V(i, j)$, is the sum of the price offers of the bids in $U(i, j)$. $U(1, j)$ will be b_1 if b_1 requests no more than j units, and otherwise will be the empty set. A recursive definition of $U(i, j)$ is given in Figure 6.6. This dynamic programming procedure is polynomial, and yields the desired result; the optimal allocation of k units is given by $U(\ell, k)$. Set $singleton_\gamma(k) = U(\ell, k)$, $1 \leq k \leq units(\gamma)$.

```
$e(b_i) > j$:
    $U(i,j) = U(i-1,j)$;
$e(b_i) = j$:
    if $p(b_i) > V(i-1,j)$
        then $U(i,j) = b_i$.
    Else $U(i,j) = U(i-1,j)$.
$e(b_i) < j$:
    if $V(i-1,j) \geq p(b_i) + V(i-1,j-e(b_i))$
        then $U(i,j) = U(i-1,j)$.
    Else $U(i,j) = b_i \cup U(i-1,j-e(b_i))$.
```

Figure 6.6: Singleton Pre-processing Algorithm

```
Let $v(j)$=0
Let $m(j)$=0
For $i = 1$ to $|L_\sigma(j)|$:
    if $m(j) < units_j(\pi)$ and $L_\sigma(j)_k \cap \pi = \emptyset$ then
        let $d := min(e(L_\sigma(j)_i)_j, units(j) - m(j))$
        $m(j) = m(j) + d$
        $v(j) = v(j) + a(L_\sigma(j)_i) \cdot d$
```

Figure 6.7: Upper Bound Algorithm

6.2.4 Upper Bound

CAMUS's upper bound function generalizes the CASS upper bound function described in Section 6.1.4, considering average price per unit rather than average price per good. Let π be the current allocation; recall that $units_i(\pi)$ denotes the number of available units of good j. Redefine $a(b_i) = \frac{p(b_i)}{\Sigma_{j=1}^{m} e(b_i)_j}$: the average price per unit of bid b_i. Define $L_\sigma(j)$ as before. Let $|L_\sigma(j)|$ denote the number of elements in $L_\sigma(j)$, and let $L_\sigma(j)_k$ denote the k^{th} element of $L_\sigma(j)$. $v(j)$ is determined by the algorithm given in Figure 6.7.

Theorem 6.2 *Let $B^* = \{b_1^*, b_2^*, \ldots, b_s^*\}$ be the bids in an optimal allocation. Then, $r^* = \Sigma_{b \in B^*} p(b) \leq \Sigma_{1 \leq j \leq m} v(j)$.*

Proof. Consider the bid $b^* \in B^*$. Then, $p(b^*) = \Sigma_{1 \leq j \leq m} a(b^*) e(b^*)_j$. Hence, $r^* = \Sigma_{b \in B^*} p(b) = \Sigma_{b \in B^*} \Sigma_{1 \leq j \leq m} a(b) e(b)_j$. By changing the order of summation we get that $r^* = \Sigma_{1 \leq j \leq m} \Sigma_{b \in B^*} a(b) e(b)_j$. Notice that, given a particular j, the contribution of bid b to $\Sigma_{b \in B^*} a(b) e(b)_j$ is $a(b) e(b)_j$. Recall now that $v(j)$ has been constructed from the set of all bids that refer to good j by choosing the maximal available units of good j from the bids in $L_\sigma(j)$, where these bids are sorted according to the average price per unit of good. Hence, we get $v(j) \geq \Sigma_{b \in B^*} a(b)e(b)_j$. Given that the above holds for every good j, this implies that $\Sigma_{1 \leq j \leq m} v(j) \geq \Sigma_{b \in B^*} p(b)$, as requested. ■

6.2.5 Heuristics

We must update our good- and bid-ordering heuristics for the multi-unit case. For our good-ordering heuristic we compute $score_i = \frac{numbids_i \cdot units(i)}{avgunits_i}$, where $numbids_i$ is the number of bids that request good i and $avgunits_i$ is the average number of *total* units (*i.e.*, not just units of good i) requested by these bids. The intuition here is similar to the intuition described in Section 6.1.6:

- We want to minimize the number of bids in low-order bins, to minimize early branching and thus to make each individual prune more effective.
- We want to minimize the number of units of goods corresponding to low-order bins, so that we will more quickly move beyond the first few bins. As a result, the pruning function will be able to take into account more contextual information.
- We want to maximize the total number of units requested by bids in low-order bins. Taking these bids moves us more quickly towards the leaves of the search tree, again providing the pruning function with more contextual information.

Given current partial allocation π, we sort bids in a given bin in descending order of $score(b_j)$, where $score(b_j) = \frac{p(b_j)}{units(b_j)} + h(\pi \cup b_j)$, which is a direct generalization of the heuristic discussed in Section 6.1.7.

6.3 Conclusions

We have presented CASS and CAMUS, algorithms that solve the single-unit and multi-unit WDPs respectively. In the next chapter, we discuss test data for evaluating such algorithms. We go on to evaluate CASS using this data in Chapter 8.

```
Process dominated bids.
Determine an ordering on the goods, according to the good-ordering
 heuristic.
Using the dynamic programming technique, determine the optimal
 combination of singleton bids totaling $1 \ldots units(j)$ for each good
 $j$.
Partition all non-singleton bids into bins, according to the good
 ordering.
Precompute pruning information for each bin.
Set $i = 1$ and $\pi = \{\}$.
Recursive entry point:
    For $j$ = 1 ...number of bids in the current subbin of
     $bin_i$.
        $\pi = \pi \cup b_j$.
        If $(p(\pi) + c(g(\pi)) \leq p(\pi_{best}))$ backtrack.
        If $(p(\pi) + h(\pi) \leq p(\pi_{best}))$ backtrack.
        If $(units(\pi) = \sum_{\gamma \in G} units(\gamma))$ record $\pi$ if it is the best;
         backtrack.
        Set $i$ to the index of the lowest-order good in $\pi$ where
         $units_i(\pi) < units(i)$. ($i$ may or may not change)
        Construct a new subbin based on the previous subbin of $bin_i$
         (which is $bin_i$ itself if $i$ changed above):
            Include all $b_k$ from current subbin, where $k > j$.
            Include all dominated bids associated with $b_j$.
            Include $singleton_i(units(i) - units_i(\pi))$.
            Sort the subbin according to the subbin-ordering
             heuristic.
            Recurse to the recursive entry point, above, and search
             this new subbin.
        $\pi = \pi \setminus b_j$.
    End For
Return the optimal allocation: $\pi_{best}$.
```

Figure 6.8: CAMUS Pseudocode

Chapter 7

Benchmarks for the Combinatorial Auction Winner Determination Problem

There has been much interest in the construction of algorithms for determining the winners of a general combinatorial auction. Before it is possible to decide the extent to which such an effort has been successful, however, it is necessary to evaluate the algorithm using some sort of test data. Since general CA's have never been widely held, there is little data recording the bidding behavior of real bidders upon which such test data may be built. In the absence of such natural benchmarks, we are left only with the option of generating artificial data that is representative of the sort of scenarios one is likely to encounter. This chapter describes such a test suite.

7.1 Past Work on Testing CA Algorithms

7.1.1 Experiments with Human Subjects

One approach to experimental work on combinatorial auctions uses human subjects. These experiments assign valuation functions to subjects, then have them participate in auctions using various mechanisms [Banks *et al.*, 1989; Ledyard *et al.*, 1997;

DeMartini *et al.*, 1998]. Such tests can be useful for understanding how real people bid under different auction mechanisms; however, they are less suitable for evaluating the mechanisms' computational characteristics. In particular, this sort of test is only as good as the subjects' valuation functions, which in the above papers were hand-crafted. As a result, this technique does not easily permit arbitrary scaling of the problem size, a feature that is important for characterizing an algorithm's performance. In addition, this method relies on relatively naive subjects to behave rationally given their valuation functions, which may be unreasonable when subjects are faced with complex and unfamiliar mechanisms.

7.1.2 Particular Problems

A parallel line of research has examined particular problems to which CA's seem well suited. For example, researchers have considered auctions for the right to use railroad tracks [Brewer & Plott, 1996], real estate [Quan, 1994], pollution rights [Ledyard & Szakaly, 1994], airport time slot allocation [Rassenti *et al.*, 1982] and distributed scheduling of machine time [Wellman *et al.*, 1998]. Most of these papers do not suggest holding an unrestricted general CA, presumably because of the computational obstacles. Instead, they tend to discuss alternative mechanisms that are tailored to the particular problem. None of them proposes a method of generating test data, nor does any of them describe how the problem's difficulty scales with the number of bids and goods. However, they still remain useful to researchers interested in general CA's because they give specific descriptions of problem domains to which CA's may be applied.

7.1.3 Artificial Distributions

A number of researchers have proposed algorithms for determining the winners of general CA's. In the absence of test suites, some have suggested novel bid generation techniques, parameterized by number of bids and goods [Sandholm, 1999; Fujishima *et al.*, 1999; Boutilier *et al.*, 1999; de Vries & Vohra, 2003]. (Other researchers have used one or more of these distributions, *e.g.*, [Parkes, 1999; Sandholm *et al.*, 2001], while still others have refrained from testing their algorithms altogether, *e.g.*, [Nisan, 2000; Lehmann *et al.*, 1999].) Parameterization represents a step forward, making it possible to describe performance with respect to the problem size. However, there are several ways in which each of these bid generation techniques falls short of realism, concerning the selection of which goods and how many goods to request in a bundle, what price to offer for the bundle, and which bids to combine in an XOR'ed set. More fundamentally, however, all of these approaches suffer from failing to model bidders explicitly, and from attempting to represent an economic situation with an non-economic model.

Which goods

First, each of the distributions for generating test data discussed above has the property that all bundles of the same size are equally likely to be requested. This assumption is clearly violated in almost any real-world auction: most of the time, certain goods (for which "natural" complementarities exist) will be more likely to appear together than others.

Number of goods

Likewise, each of the distributions for generating test data determines the number of goods in a bundle completely independently from determining *which* goods appear in the bundle. While this assumption may appear more reasonable, there are many domains in which the expected number of items in a bundle will be related to which goods it contains. (For example, people buying computers will tend to make long combinatorial bids, requesting monitors, printers, etc., while people buying refrigerators will tend to make short bids.)

Price

Next, there are problems with the schemes for generating price offers used by all four techniques. Pricing is especially crucial: if prices are not chosen carefully then an otherwise hard distribution can become computationally easy.

In Sandholm [Sandholm, 1999] prices are drawn randomly from either $[0, 1]$ or from $[0, g]$, where g is the number of goods requested. The first method is clearly unreasonable (and computationally trivial) since price is unrelated to the number of goods in a bid—note that a bid for many goods and for a small subset of the same bid will have exactly the same price on expectation. The second is better, but has the disadvantage that average and range are parameterized by the same variable.

In Boutilier et al.[Boutilier *et al.*, 1999] prices of bids are distributed normally with mean 16 and standard deviation 3, giving rise to the same problem as the $[0, 1]$ case above.

In Fujishima et al.[Fujishima *et al.*, 1999] prices are drawn from $[g(1 - d), g(1 + d)]$, $d = 0.5$. While this scheme avoids the problems described above, prices are simply additive in g and are unrelated to *which* goods are requested in a bundle, both unrealistic assumptions in some domains.

More fundamentally, Anderson et al.[Anderson *et al.*, 2000] note a critical pricing problem that arises in several of the schemes discussed above. As the number of bids to be generated becomes large, a given short bid will be drawn much more frequently than a given long bid. Since the highest-priced bid for a bundle dominates all other bids for the same bundle, short bids end up being much more competitive. Indeed, it is pointed out that for extremely large numbers of bids a good approximation to the optimal solution is simply to take the best singleton bid for each good. One solution to this problem is to guarantee that only the first bid for each bundle will be retained. However, this solution has the drawback that it is unrealistic: different real bidders *are* likely to place bids on some of the same bundles.

Another solution to this problem is to make bundle prices superadditive in the number of goods they request—an assumption that may also be reasonable in many CA domains. A similar approach is taken by deVries and Vohra [de Vries & Vohra, 2003], who make the price for a bid a quadratic function of the prices of bids for subsets. For some domains this pricing scheme may result in too large an increase in price as a function of bundle length. The distributions presented in this chapter will include a pricing scheme that may be configured to be superadditive or subadditive in bundle length, where appropriate, parameterized to control how rapidly the price offered increases or decreases as a function of bundle length.

XOR bids

Finally, while most of the bid-generation techniques discussed above permit bidders to submit sets of bids XOR'ed together, they have no way of generating meaningful sets of such bids. As a consequence the computational impact of XOR'ed bids has been very difficult to characterize.

7.2 Generating Realistic Bids

While the lack of standardized, realistic test cases does not make it impossible to evaluate or compare algorithms, it does make it difficult to know what magnitude of real-world problems each algorithm is capable of solving, or what features of real-world problems each algorithm is capable of exploiting. This second ambiguity is particularly troubling: it is likely that algorithms would be designed *differently* if they took the features of more realistic[1] bidding into account.

7.2.1 Prices, price offers and valuations

The term "price" has traditionally been used by researchers constructing artificial distributions to describe the amount offered for a bundle. However, this term really refers to the amount a bidder is made to pay for a bundle, which is of course mechanism-specific and is often not the same as the amount offered. Indeed, it is impossible to model bidders' price offers at all without committing to a particular auction mechanism. In the distributions described in this chapter, we will assume a sealed-bid incentive-compatible mechanism, where the price offered for a bundle is equal to the bidder's valuation. Hence, in the rest of this chapter, we will use the

[1]There does exist a body of previous work characterizing hard cases for weighted set packing, which is of course equivalent to the combinatorial auction problem. Real-world bidding is likely to exhibit various regularities, however, as discussed throughout this chapter. A data set designed to include the same regularities may be more useful for predicting the performance of an algorithm in a real-world combinatorial auction.

terms *price offer* and *valuation* interchangeably. Researchers wanting to model bidding behavior in other mechanisms could transform the valuation generated by our distributions according to bidders' equilibrium strategies in the new mechanism.

7.2.2 The CATS suite

In this chapter we present CATS (Combinatorial Auction Test Suite), a suite of distributions for modeling realistic bidding behavior. This suite is grounded in previous research on specific applications of combinatorial auctions, as described in section 7.1.1 above. At the same time, all of our distributions are parameterized by number of goods and bids, facilitating the study of algorithm performance. This suite represents a move beyond previous work on modeling bidding in combinatorial auctions because we provide an economic motivation for both the contents and the valuation of a bundle, deriving them from basic bidder preferences. In particular, in each of our distributions:

- Certain goods are more likely to appear together than others.
- The number of goods appearing in the bundle is often related to which goods appear in the bundle.
- Valuations are related to which goods appear in the bundle. Where appropriate, valuations can be configured to be subadditive, additive or superadditive in the number of goods requested.
- Sets of XOR'ed bids are constructed in meaningful ways, on a per-bidder basis.

CATS suite contains a legacy section including all bid generation techniques described above, so that new algorithms may easily be compared to previously-published results. More information on the test suite, including executable versions of our distributions for Linux and Windows may be found on the CATS website [2000].

In section 7.3, below, we present distributions based on five real-world situations. For most of our distributions, the mechanism for generating bids requires first building a graph representing adjacency relationships between goods. Later, the mechanism uses the graph, generated in an economically-motivated way, to derive complementarity properties between goods and substitutability properties for bids. Of the five real-world situations we model, the first three concern complementarity based on adjacency in (physical or conceptual) space, while the final two concern complementarity based on correlation in time. Our first example (section 7.3.1) models shipping, rail and bandwidth auctions. Goods are represented as edges in a nearly planar graph, with agents submitting an XOR'ed set of bids for paths connecting two nodes. Our second example (section 7.3.2) models an auction of real estate, or more generally of any goods over which two-dimensional adjacency is the basis of complementarity. Again the relationship between goods is represented by a graph, in this case strictly planar. In Section 7.3.3 we relax the planarity assumption from the previous example in order to model arbitrary complementarities between discrete goods such as electronics parts or collectables. Our fourth example (section 7.3.4) concerns the matching of time-slots for a fixed number of different goods; this case applies to airline take-off and landing rights auctions. In Section 7.3.5 we discuss the generation of bids for a distributed job-shop scheduling domain, and also its application to power generation auctions. Finally, in Section 7.3.7, we provide a legacy suite of bid generation techniques, including all those discussed in Section 7.1.3 above.

In the description of the distributions that follow, let $rand(a, b)$ represent a real number drawn uniformly from $[a, b]$. Let $rand\_int(a, b)$ represent a random integer drawn uniformly from the same interval. With respect to a given graph, let $e(x, y)$ represent the proposition that an edge exists between nodes x and y. All of the distributions presented here are parameterized by the number of goods (num_goods) and number of bids (num_bids).

7.3 CATS in Detail

7.3.1 Paths in Space

There are many real-world problems that involve bidding on paths in space. Generally, this class may be characterized as the problem of purchasing a connection between two points. Examples include truck routes [Sanholm, 1993], natural gas pipeline networks [Rassenti *et al.*, 1994], network bandwidth allocation, and the right to use railway tracks [Brewer & Plott, 1996].[2] In particular, spatial path problems consist of a set of points and accessibility relations between them. Although the distribution we propose may be configured to model bidding in any of the above domains, we will use the railway domain as our motivating example since it is both intuitive and well-understood.

More formally, we will represent this railroad auction by a graph in which each node represents a location on a plane, and an edge represents a connection between locations. The goods at auction are therefore the edges of the graph, and bids request a set of edges that form a path between two nodes. We assume that no bidder will desire more than one path connecting the same two nodes, although the bidder may value each path differently.

Building the Graph

The first step in modeling bidding behavior for this problem is determining the graph of spatial and connective relationships between cities. One approach would be to use an actual railroad map, which has the advantage that the resulting graph would be unarguably realistic. However, it would be difficult to find a set of real-world maps that could be said to exhibit a similar sort of connectivity and would encompass substantial variation in the number of cities. Since scalability of input data is of great importance to the testing of new CA algorithms, we have chosen to propose generating such graphs randomly. Figure 7.1 shows a representative example of a graph generated using our technique.

We begin with *num_cities* nodes randomly placed on a plane. We add edges to this graph, G, starting by connecting each node to a fixed number of its nearest neighbors.

[2]Electric power distribution is a frequently discussed real world problem which seems superficially similar to the problems discussed here. However, many of the complementarities in this domain arise from physical laws governing power flow in a network. Consideration of these laws becomes very complex in networks of interesting size. Also, because these laws are taken into account during the construction of power networks, the networks themselves are difficult to model using randomly generated graphs. For these reasons, we do not attempt to model this domain.

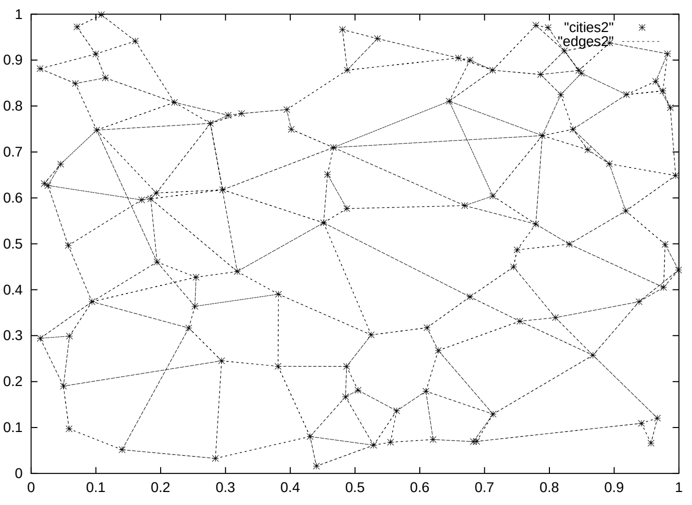


Figure 7.1: Sample Railroad Graph

Next, we iteratively consider random pairs of nodes and examine the shortest path connecting them, if any. To compare, we also compute various alternative paths that would require one or more edges to be added to the graph, given a penalty proportional to distance for adding new edges. (We do this by considering a complete graph C, an augmentation of G with new edges weighted to reflect the distance penalty.) If the shortest path involves new edges—despite the penalty—then the new edges (without penalty) are added to G, and replace the existing edges in C. This process models our simplifying assumption that there will exist uniform demand for shipping between any pair of cities, though of course it does not mimic the way new links would actually be added to a rail network. The process continues until slightly more edges have been created than the number of goods in the auction being modeled. (This is achieved by the “1.05” in the first line of Figure 7.2.) The reason more edges than are necessary are created is that some edges will not appear in bids, hence should not be considered as goods.

Our technique produces slightly non-planar graphs—graphs on a plane in which edges occasionally cross at points other than nodes. We consider this to be reasonable, as the same phenomenon may be observed in real-world rail lines, highways, network wiring, etc. Determining the “reasonableness” of a graph is of course a subjective task

```
Let $num\_cities = num\_goods \div edge\_density \times 1.05$
Randomly place nodes (cities) on a unit box
Connect each node to its $initial\_connections$ nearest neighbors
While $num\_edges < num\_cities \times edge\_density$
    $C = G$
    For every pair of nodes $n_1, n_2 \in G$ where $\neg e(n_1, n_2)$
        Add an edge to $C$ of length
          $building\_penalty \cdot Euclidean\_distance(n_1, n_2)$
    End For
    Choose two nodes at random, and find the shortest path between
      them in $C$
    If shortest path uses edges that do not exist in $G$
        For every such pair of nodes $n_1, n_2 \in G$ add an edge to $G$ with
          length $Euclidean\_distance(n_1, n_2)$
    End If
End For
```

Figure 7.2: Graph-Building Technique

unless more quantitative metrics are used to assess quality; we see the identification and application of such metrics (for this and other distributions) as an important topic for future work.

Generating Bids

Given a map of cities and the connectivity between them, there is the orthogonal problem of modeling bidding itself. We propose a method which generates a set of substitutable bids from a hypothetical agent's point of view. We start with the value to an agent for shipping from one city to another and with a shipping cost which we make equal to the Euclidean distance between the cities. We then place XOR bids on all paths on which the agent would make a profit (*i.e.*, those paths where $utility - cost > 0$). The path's value is random, in (parameterized) proportion to the Euclidean distance between the chosen cities. Since the shipping cost is the Euclidean distance between two cities, we use this as the lower bound for value as well, since only bidders with such valuations would actually place bids.

There will occasionally be edges that are useful only for shipping directly between the two cities they connect. These edges are clearly unrealistic; also, because they will only be selected for singleton bids, they will not increase the size of the search space. In fact, the same argument can be made about any small groups of bids that do not conflict with any bids outside of the group: they are better modeled as a separate auction, and contribute very little to the difficulty of the winner determination problem. Recall that we started out with more goods than we want to generate. At some point in the bid generation process—usually before we have generated all of the bids—the total number of goods requested across all bids will meet our target. At this point we check for edges that are used only in singleton bids or isolated groups of bids, and delete those bids. Once we reach the target number of goods without deleting any bids, we delete all goods that are uninvolved in the bids we have generated so far, and continue with bid generation.

If we reach our target number of bids *without* making use of the target number of goods, we must restart graph generation. When restarting, we increase the number of goods and cities to increase the expected number of different goods used as a fraction of bids generated.

Note that this distribution, and indeed all others presented in this chapter, may generate slightly more than *num_bids* bids. In our experience CA optimization algorithms tend not to be highly sensitive in the number of bids, so we judged it more important to build economically sensible sets of substitutable bids. When generating a precise number of bids *is* important, an appropriate number of bids may be removed after all bids have been generated so that the total will be met exactly.

Note that 1 is used as a lower bound for d because any bidder with $d < 1$ would find no profitable paths and therefore would not bid.

Multi-Unit Extensions: Bandwidth Allocation, Commodity Flow

This model may also be used to generate realistic data for multi-unit CA problems such as network bandwidth allocation and general commodity flow. The graph may be created as above, but with a number of units (capacity) assigned to each edge. Likewise, the bidding technique remains unchanged except for the assignment of a number of units to each bid.

```
$finished = true$
Do
    While $num\_generated\_bids < num\_bids$
        Randomly choose two nodes, $n_1$ and $n_2$
        $d = rand(1, shipping\_cost\_factor)$
        $cost = Euclidean\_distance(city_1, city_2)$
        $value = d \cdot Euclidean\_distance(city_1, city_2)$
        Make XOR bids of $value - cost$ on every path from $city_1$ to $city_2$
          having $cost < value$
        If there are more than $max\_bid\_set\_size$ such paths, bid on the
          $max\_bid\_set\_size$ paths that maximize $value - cost$.
        If number of goods receiving bids $\geq num\_goods$
            remove isolated singleton bids and isolated bid groups
            remove from the city map all edges that do not
              participate in any bid
        End If
    End While
    If number of goods receiving bids $< num\_goods$
        delete all bids
        delete graph
        $num\_cities = num\_cities + 1$
        run graph generation
        $finished = false$
    End While
While $\neg finished$
```

Figure 7.3: Bid-Generation Technique

7.3.2 Proximity in Space

There is a second broad class of real-world problems in which complementarity arises from adjacency in two-dimensional space. An intuitive example is the sale of adjacent pieces of real estate [Quan, 1994]. Another example is drilling rights, where it is much cheaper for an oil company to drill in adjacent lots than in lots that are far from each other. In this section, we first propose a graph-generation mechanism that builds a model of adjacency between goods, and then describe a technique for generating realistic bids on these goods. Note that in this section nodes of the graph represent the goods at auction, while edges represent the adjacency relationship.

```
Place nodes at integer vertices $(i, j)$ in a plane, where
 $1 \leq i, j \leq \lceil\sqrt(num\_goods)\rceil$
For each node $n$
    If $n$ is on the edge of the map
        Connect $n$ to as many hv-neighbors as possible
    Else
        If $rand(0, 1) \leq three\_prob$
            Connect $n$ to a random set of three of its four
             hv-neighbors
        Else
            Connect $n$ to all four of its hv-neighbors
        While $rand(0, 1) \leq additional\_neighbor$
            Connect $g$ to one of its d-neighbors, provided that the
             new diagonal edge will not cross another diagonal edge
        End While
End For
```

Figure 7.4: Graph-Building Technique

Building the Graph

There are a number of ways we could build an adjacency graph. The simplest would be to place all the goods (locations, nodes) in a grid, and connect each to its four neighbors. We propose a slightly more complex method in order to permit a variable number of neighbors per node (corresponding, for example, to non-rectangular pieces of real estate). As above we place all goods on a grid, but with some probability we omit a connection between goods that would otherwise represent vertical or horizontal adjacency, and with some probability we introduce a connection representing diagonal adjacency. (We call horizontally- or vertically-adjacent nodes *hv-neighbors* and diagonally-adjacent nodes *d-neighbors*.)

Figure 7.5 shows a sample real estate graph, generated by the technique described in Figure 7.4. Nodes of the graph are shown with asterisks, while edges are represented by solid lines. The dashed lines show one set of property boundaries that would be represented by this graph. Note that one node falls inside each piece of property, and that two pieces of property border each other iff their nodes share an edge.

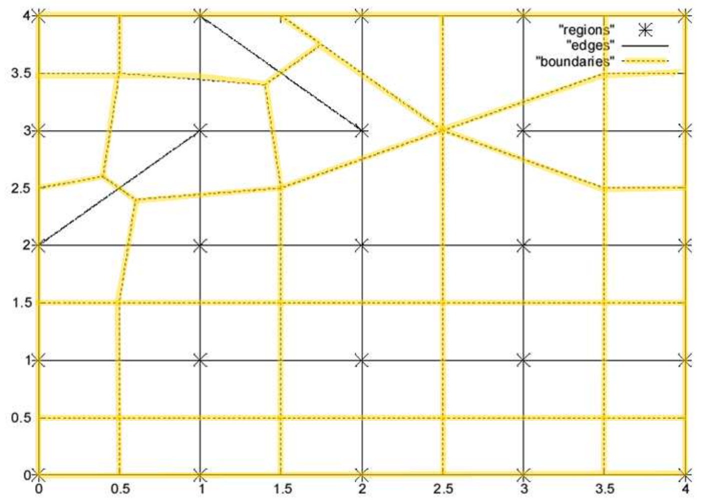


Figure 7.5: Sample Real Estate Graph

Generating Bids

To model realistic bidding behavior, we generate a set of common values for each good, and private values for each good for each bidder. The common value represents the appraised or expected resale value of each individual good. The private value represents how much one particular bidder values that good, as an offset to the common value (*e.g.*, a private value of 0 for a good represents agreement with the common value). These private valuations describe a bidder's preferences, and so they are used to determine both a value for a given bid and the likelihood that a bidder will request a bundle that includes that good. There are two additional components to each bidder's preferences: a minimum total common value, and a budget. The former reflects the idea that a bidder may only wish to acquire goods of a certain recognized value. The latter reflects the fact that a bidder may not be able to afford every bundle that is of interest to him.

To generate bids, we first add a random good, weighted by a bidder's preferences, to the bidder's bid. Next, we determine whether another good should be added by drawing a value uniformly from [0,1], and adding another good if this value is smaller than a threshold. This is equivalent to drawing the number of goods in a bid from a decay distribution.[3][4] We must now decide which good to add. First we allow a small chance that a new good will be added uniformly at random from the set of goods, without the requirement that it be adjacent to a good in the current bundle B . (This permits bundles requesting unconnected regions of the graph: for example, a hotel company may only wish to build in a city if it can acquire land for two hotels on opposite sides of the city.) Otherwise, we select a good from the set of nodes bordering the goods in B. The probability that some adjacent good n_1 will be added depends on how many edges n_1 shares with the current bundle, and on the bidder's relative private valuations for n_1 and n_2. For example, if nodes n_1 and n_2 are each connected to B by one edge, and the private valuation for n_1 is twice that for n_2 then the probability of adding n_1 to B, $p(n_1)$, is $2p(n_2)$. Further, if n_1 has 3 edges to nodes in B while n_2 is connected to B by only 1 edge, and the goods have equivalent private values, then $p(n_1) = 3p(n_2)$. Once we have determined all the goods in a bundle we set the price offered for the bundle, which depends on the sum of common and private valuations for the goods in the bundle, and also includes a function that is superadditive (with our parameter settings) in the number of goods.[5] Finally, we generate additional bids that are substitutable for the original bid, with the constraint that each bid in the set requests at least one good from the original bid.

Spectrum Auctions

A related problem is the auction of radio spectrum, in which a government sells the right to use specific segments of spectrum in different geographical areas[Plott & Cason, 1996; Ausubel *et al.*, 1997]. It is possible to approximate bidding behavior in spectrum auctions by making the assumption that all complementarity arises from

[3]We use Sandholm's [Sandholm, 1999] term "decay" here, though the distribution goes by various names—for a description of the distribution please see Section 7.3.6.

[4]There are two reasons we use a decay distribution here. First, we expect that more bids will request small bundles than large bundles. Second, we require a distribution where the expected bundle size is (relatively) unaffected by changes in the total number of goods.

[5]Recall the discussion in Section 7.1.3 motivating the use of superadditive valuations.

```
Routine Add_Good_to_Bundle(bundle $B$)
    If $rand(0,1) \leq jump\_prob$
        Add a good $g \notin b$ to $B$, chosen uniformly at random
    Else
        Compute $s = \sum_{x \notin B, y \in B, e(x,y)} pn(x)$ [$pn()$ is defined below]
        Choose a random node $x \notin B$ from the distribution
          $\sum_{y \in B, e(x,y)} \frac{pn(x)}{s}$
        Add $x$ to $B$
    End If
End Routine
```

Figure 7.6: Add_Good_to_Bundle for Spatial Proximity

spatial proximity.[6] In this case, our spatial proximity model can also be used to generate bidding distributions for spectrum auctions. The main difference between this problem and the real estate problem is that in a spectrum auction each good may have multiple units (frequency bands) for sale. It is insufficient to model this as a multi-unit CA problem, however, if bidders have the constraint that they want the same frequency in each region.[7] Instead, the problem can be modeled with multiple distinct goods per node in the graph, and bids constructed so that all nodes added to a bundle belong to the same 'frequency'. With this method, it is also easy to incorporate other preferences, such as preferences for different types of goods. For instance, if two different types of frequency bands are being sold, one 5 megahertz wide and one 2.5 megahertz wide, an agent only wanting 5 megahertz bands could make substitutable bids for each such band in the set of regions desired (generating the bids so that the agent will acquire the same frequency in all the regions).

The scheme for generating price offers used in our real estate example may be inappropriate for the spectrum auction domain. Research indicates that while price offers will still tend to be superadditive, this superadditivity may be quadratic in the

[6]This assumption would be violated, for example, if some bidders wanted to secure the right to broadcast at the same frequency in several adjacent areas.

[7]To see why this cannot be modeled as a multi-unit CA, consider an auction for three regions with two units each, and three bidders each wanting one unit of two goods. In the optimal allocation, b_1 gets 1 unit of g_1 and 1 unit of g_2, b_2 gets 1 unit of g_2 and 1 unit of g_3, and b_3 gets 1 unit of g_3 and 1 unit of g_1. In this example there is no way of assigning frequencies to the units so that each bidder gets the same frequency in both regions.

```
For all $g, c(g) = rand(1, max\_good\_value)$
While $num\_generated\_bids < num\_bids$
    For each good, reset
      $p(g) = rand(-deviation \cdot max\_good\_value, deviation + max\_good\_value)$
    $pn(g) = \frac{p(g)+deviation \cdot max\_good\_value}{2 \cdot deviation \cdot max\_good\_value}$
    Normalize $pn(g)$ so that $\sum_g pn(g) = 1$
    $B = \{\}$
    Choose a node $g$ at random, weighted by $pn()$, and add it to $B$
    While $rand(0,1) \leq additional\_location$
        Add_Good_to_Bundle($B$)
    $value(B) = \sum_{x \in B}(c(x) + p(x)) + S(|B|)$
    If $value(B) \leq 0$ on $B$, restart bundle generation for this bidder
    Bid $value(B)$ on $B$
    $budget = budget\_factor \cdot value(B)$
    $min\_resale\_value = resale\_factor \cdot \sum_{x \in B} c(x)$
    Construct substitutable bids.  For each good $g_i \in B$
        Initialize a new bundle, $B_i = \{g_i\}$
        While $|B_i| < |B|$
            Add_Good_to_Bundle($B_i$)
        Compute $c_i = \sum_{x \in B_i} c(x)$
    End For
    Make XOR bids on all $B_i$ where $0 \leq value(B) \leq budget$ and
      $c_i \geq min\_resale\_value$.
    If there are more than $max\_substitutable\_bids$ such bundles, bid on
      the $max\_substitutable\_bids$ bundles having the largest $value$
End While
```

Figure 7.7: Bid-Generation Technique

population of the region rather than exponential in the number of regions [Ausubel *et al.*, 1997].

7.3.3 Arbitrary Relationships

Sometimes complementarities between goods will not be as universal as geographical adjacency, but some kind of regularity in the complementarity relationships between goods will still exist. Consider an auction of different, indivisible goods, *e.g.*, for semiconductor parts or collectables, or for distinct multi-unit goods such as the right to emit some quantity of two different pollutants produced by the same industrial

```
Build a fully-connected graph with one node for each good
Label each edge from $n_1$ to $n_2$ with a weight $d(n_1, n_2) = rand(0, 1)$
```

Figure 7.8: Graph-Building Technique

process. In this section we discuss a general way of modeling such arbitrary relationships.

Building the Graph

We express the likelihood that a particular pair of goods will appear together in a bundle as being proportional to the weight of the appropriate edge of a fully-connected graph. That is, the weight of an edge between n_1 and n_2 is proportional to the probability that, having only n_1 in our bundle, we will add n_2. Weights are only proportional to probabilities because we must normalize the sum of all weights from a given good to sum to 1.

Generating Bids

Our technique for modeling bidding is a generalization of the technique presented in the previous section. We choose a first good and then proceed to add goods one by one, with the probability of each new good being added depending on the current bundle. Note that, since in this section the graph is fully-connected, there is no need for the 'jumping' mechanism described above. The likelihood of adding a new good g to bundle B is proportional to $\sum_{y \in B} d(x, y) \cdot p_i(x)$. The first term $d(x, y)$ represents the likelihood (independent of a particular bidder) that goods x and y will appear in a bundle together; the second, $p_i(x)$, represents bidder i's private valuation of the good x. We implement this new mechanism by changing the routine *Add_Good_to_Bundle()*. We are thus able to use the same techniques for assigning a value to a bundle, as well as for determining other bundles with which it is substitutable.

```
Routine Add_Good_to_Bundle(bundle $B$)
    Compute $s = \sum_{x \notin b, y \in B} d(x,y) \cdot pn(x)$
    Choose a random node $x \notin B$ from the distribution $\sum_{y \in B} d(x,y) \cdot \frac{pn(x)}{s}$
    Add $x$ to $B$
End Routine
```

Figure 7.9: Routine Add_Good_to_Bundle for Arbitrary Relationships

7.3.4 Temporal Matching

We now consider real-world domains in which complementarity arises from a temporal relationship between goods. In this section we discuss matching problems, in which corresponding time slices must be secured on multiple resources. The general form of temporal matching includes m sets of resources, in which each bidder wants 1 time slice from each of $j \leq m$ sets subject to certain constraints on how the times may relate to one another (*e.g.*, the time in set 2 must be at least two units later than the time in set 3). Here we concern ourselves with the problem in which $j = 2$, and model the problem of airport take-off and landing rights. Rassenti et al. [Rassenti *et al.*, 1982] made the first study of auctions in this domain. The problem has been the topic for much other work; in particular [Grether *et al.*, 1989] includes detailed experiments and an excellent characterization of bidder behavior.

The airport take-off and landing problem arises because certain high-traffic airports require airlines to purchase the right to take off or land during a given time slice. However, if an airline buys the right for a plane to take off at one airport then it must also purchase the right for the plane to land at its destination an appropriate amount of time later. Thus, complementarity exists between certain pairs of goods, where goods are the right to use the runway at a particular airport at a particular time. Substitutable bids are different departure/arrival packages; therefore bids will only be substitutable within certain limits.

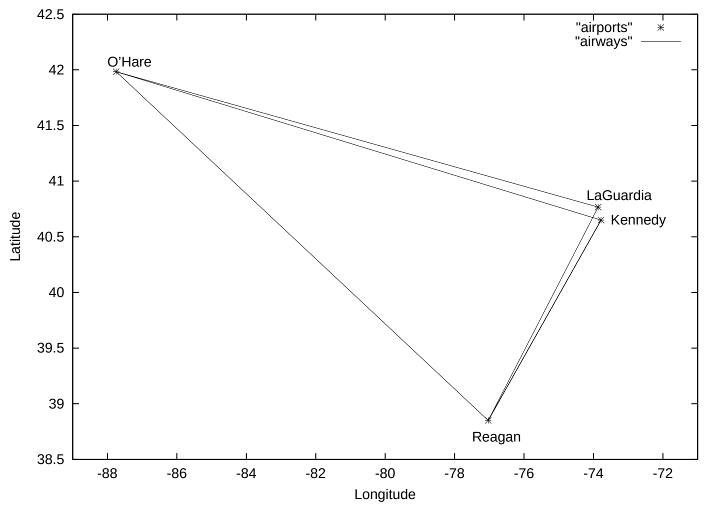


Figure 7.10: Map of Airport Locations

Building the Graph

Departing from our graph-based approach above, we ground this example in the real map of high-traffic US airports for which the Federal Aviation Administration auctions take-off and landing rights, described in [Grether *et al.*, 1989]. These are the four busiest airports in the United States: La Guardia International, Ronald Reagan Washington National, John F. Kennedy International, and O'Hare International. This map is shown below.

We chose not to use a random graph in this example because the number of bids and goods is dependent on the number of bidders and time slices at the given airports; it is not necessary to modify the number of *airports* in order to vary the problem size. Thus, $num\_cities = 4$ and $num\_times = \lfloor num\_goods/num\_cities \rfloor$.

Generating Bids

Our bidding mechanism presumes that airlines have a certain tolerance for when a plane can take off and land ($early\_takeoff\_deviation$, $late\_takeoff\_deviation$, $early\_land\_deviation$, $late\_land\_deviation$), as related to their most preferred take-off and landing times ($start\_time$, $start\_time + min\_flight\_length$). We generate bids for all bundles that fit these criteria. The value of a bundle is derived from a particular agent's utility function. We define a utility u_{max} for an agent, which corresponds to the utility the agent receives for flying from $city_1$ to $city_2$ if it receives the ideal takeoff and landing times. This utility depends on a common value for a time slot at the given airport, and deviates by a random amount. Next we construct a utility function which reduces u_{max} according to how late the plane will arrive, and how much the flight time deviates from optimal.

```
Set the average valuation for each city's airport:
  $cost(city) = rand(0, max\_airport\_value)$
Let $max\_l$ = length of longest distance between any two cities
While $num\_generated\_bids < num\_bids$
    Randomly select $city_1$ and $city_2$ where $e(city_1, city_2)$
    $l = distance(city_1, city_2)$
    $min\_flight\_length = round(longest\_flight\_length \cdot \frac{1}{max\_l})$
    $start\_time = rand\_int(1, num\_times - min\_flight\_length)$
    $dev = rand(1 - deviation, 1 + deviation)$
    Make substitutable (XOR) bids.  For
      $takeoff = max(1, start\_time - early\_takeoff\_deviation)$ to
      $min(num\_times, start\_time + late\_takeoff\_deviation)$
        For $land = takeoff + min\_flight\_length$ to
          $min(start\_time + min\_flight\_length + late\_land\_deviation, num\_times)$

            $amount\_late = min(land - (start\_time + min\_flight\_length), 0)$
            $delay = land - takeoff - min\_flight\_length$
            Bid $dev \cdot (cost(city_1) + cost(city_2)) \cdot delay\_coeff^{delay} \cdot$
              $amount\_late\_coeff^{amount\_late}$ for takeoff at time $takeoff$ at
              $city_1$ and landing at time $land$ at $city_2$
        End For
    End For
End While
```

Figure 7.11: Bid-Generation Technique

7.3.5 Temporal Scheduling

Wellman et al. [Wellman *et al.*, 1998] proposed distributed job-shop scheduling with one resource as a CA problem. We provide a distribution that mirrors this problem.

While there exist many algorithms for solving job-shop scheduling problems, the distributed formulation of this problem places it in an economic context. Wellman et al. describe a factory conducting an auction for time-slices on some resource. Each bidder has a job requiring some amount of machine time, and a deadline by which the job must be completed. Some jobs may have additional, later deadlines which are less desirable to the bidder and so for which the bidder is willing to pay less.

Generating Bids

In the CA formulation of this problem, each good represents a specific time slice. Two bids are substitutable if they constitute different possible schedules for the same job. We determine the number of deadlines for a given job according to a decay distribution, and then generate a set of substitutable bids satisfying the deadline constraints. Specifically, let the set of deadlines of a particular job be $d_1 < \cdots < d_n$ and the value of a job completed by d_1 be v_1, superadditive in the job length. We define the value of a job completed by deadline d_i as $v_i = v_1 \cdot \frac{d_1}{d_i}$, reflecting the intuition that the decrease in value for a later deadline is proportional to its 'lateness'. Like Wellman et al., we assume that all jobs are eligible to be started in the first time slot. Our formulation of the problem differs in only one respect—we consider only allocations in which jobs receive continuous blocks of time. However, this constraint is not restrictive because for any arbitrary allocation of time slots to jobs there exists a new allocation in which each job receives a continuous block of time and no job finishes later than in the original allocation. (This may be achieved by numbering the winning bids in increasing order of scheduled end time, and then allocating continuous time-blocks to jobs in this order. Clearly no job will be rescheduled to finish later than its original scheduled time.) Note also that this problem cannot be translated to a trivial one-good multi-unit CA problem because jobs have different deadlines.

7.3.6 Legacy Distributions

To aid researchers designing new CA algorithms by facilitating comparison with previous work, CATS includes the ability to generate bids according to all previous

```
While $num\_generated\_bids < num\_bids$
    $l = rand\_int(1, max\_length)$
    $d_1 = rand\_int(l, num\_goods)$
    $dev = rand(1 - deviation, 1 + deviation)$
    $cur\_max\_deadline = 0$
    $new\_d = d_1$
    To generate substitutable (XOR) bids.  Do
        Make bids with price offered $= dev \cdot l^{1+additivity} \cdot d_1/new\_d$ for all
          blocks $[start, end]$ where $start \geq 1$, $end \leq new\_d$,
          $end > cur\_max\_deadline$, $end - start = l$
        $cur\_max\_deadline = new\_d$
        $new\_d = rand\_int(cur\_max\_deadline + 1, num\_goods)$
    While $rand(0, 1) \leq prob\_additional\_deadline$
End While
```

Figure 7.12: Bid-Generation Technique

published test distributions of which we are aware, that are able to scale with the number of goods and bids. Each of these distributions may be seen as an answer to three questions: what number of goods to request in a bundle, which goods to request, and the price offered for a bundle. We begin by describing different techniques for answering each of these three questions, and then show how they have been combined in previously published work.

Number of Goods

Uniform: Uniformly distributed on $[1, num\_goods]$

Normal: Normally distributed with $\mu = \mu\_goods$ and $\sigma = \sigma\_goods$

Constant: Fixed at *constant_goods*

Decay: Starting with 1, repeatedly increment the size of the bundle until $rand(0, 1)$ exceeds α

Binomial: Request n goods with probability $p^n(1-p)^{num\_goods-n}\binom{num\_goods}{n}$

Exponential: Request n goods with probability $C \exp^{-n/q}$

Which Goods

Random: Draw n goods uniformly at random from the set of all goods, without replacement[8]

Price Offer

Fixed Random: Uniform on $[low\_fixed, hi\_fixed]$
Linear Random: Uniform on $[low\_linearly \cdot n, hi\_linearly \cdot n]$
Normal: Draw from a normal distribution with $\mu = \mu\_price$ and $\sigma = \sigma\_price$
Quadratic[9]**:** For each good k and each bidder i set the value $v_k^i = rand(0,1)$; then i's price offer for a set of goods S is $\sum_{k \in S} v_k^i + \sum_{k,q} v_k^i v_q^i$

7.3.7 Previously Published Distributions

The following is a list of the distributions used in all published tests of which we are aware. In each case we describe first the method used to choose the number of goods, followed by the method used to choose the price offer. In all cases the 'random' technique was used to determine which goods should be requested in a bundle. Each case is labeled with its corresponding CATS legacy suite number; very similar distributions are given similar numbers and identical distributions are given the same number.
[**L1**] *Sandholm*: Uniform, fixed random with $low\_fixed = 0$, $hi\_fixed = 1$
[**L1a**] *Anderson et al.*: Uniform, fixed random with $low\_fixed = 0$, $hi\_fixed = 1000$
[**L2**] *Sandholm*: Uniform, linearly random with $low\_linearly = 0$, $hi\_linearly = 1$
[**L2a**] *Anderson et al.*: Uniform, linearly random with $low\_linearly = 500$, $hi\_linearly = 1500$
[**L3**] *Sandholm*: Constant with $constant\_goods = 3$, fixed random with $low\_fixed =$

[8] Although in principle the problem of *which* goods to request could be answered in many ways, all legacy distributions of which we are aware use this technique.

[9] DeVries and Vohra [de Vries & Vohra, 2003] briefly describe a more general version of this price offer scheme, but do not describe how to set all the parameters (*e.g.*, defining which goods are complementary); hence we do not include it here. Quadratic price offers may be particularly applicable to spectrum auctions; see [Ausubel *et al.*, 1997].

0, $hi\_fixed = 1$

[**L3**] *deVries and Vohra*: Constant with *constant_goods* = 3, fixed random with $low\_fixed = 0$, $hi\_fixed = 1$

[**L4**] *Sandholm*: Decay with $\alpha = 0.55$, linearly random with *low_linearly* = 0, *hi_linearly* = 1

[**L4**] *deVries and Vohra*: Decay with $\alpha = 0.55$, linearly random with *low_linearly* = 0, *hi_linearly* = 1

[**L4a**] *Anderson et al.*: Decay with $\alpha = 0.55$, linearly random with *low_linearly* = 1, *hi_linearly* = 1000

[**L5**] *Boutilier et al.*: Normal with *μ_goods* = 4 and *σ_goods* = 1, normal with *μ_price* = 16 and *σ_price* = 3

[**L6**] *Fujishima et al.*: Exponential with $q = 5$, linearly random with *low_linearly* = 0.5, *hi_linearly* = 1.5

[**L6a**] *Anderson et al.*: Exponential with $q = 5$, linearly random with *low_linearly* = 500, *hi_linearly* = 1500

[**L7**] *Fujishima et al.*: Binomial with $p = 0.2$, linearly random with *low_linearly* = 0.5, *hi_linearly* = 1.5

[**L7a**] *Anderson et al.*: Binomial with $p = 0.2$, linearly random with *low_linearly* = 500, *hi_linearly* = 1500

[**L8**] *deVries and Vohra*: Constant with *constant_goods* = 3, quadratic

Parkes [Parkes, 1999] used many of the test sets described above (particularly those described by Sandholm and Boutilier et al.), but tested with fixed numbers of goods and bids rather than scaling these parameters.

Since the publication of [Leyton-Brown *et al.*, 2000a], the CATS distributions have also been widely used, for example by [Sandholm *et al.*, 2001; Gonen & Lehmann, 2001; Gonen & Lehmann, 2000; Holte, 2001; Schuurmans *et al.*, 2001; Kastner *et al.*, 2002; Zurel & Nisan, 2000].

7.4 Tuning Distributions

7.4.1 Removing Dominated Bids

For the WDP, it is well known that problems become harder as the number of goods and bids increases.[10] For this reason, researchers have traditionally reported the performance of their WDP algorithms in terms of the number of bids and goods of the input instances. While it is easy to fix the number of goods, holding the number of bids constant is not as straightforward as it might appear. Most special-purpose algorithms make use of a polynomial-time preprocessing step which removes bids that are strictly dominated by one other bid. More precisely, bid i is dominated by bid j if the goods requested by i are a (non-strict) superset of the goods requested by j, and the price offer of i is smaller than or equal to the price offer of j. (This is similar in flavor to the use of arc-consistency as a preprocessing step for a CSP or weighted CSP problem.) It is thus possible for the size of problems given as input to the WDP algorithm to vary even if all generated instances had the same number of bids.

It is not obvious whether this domination procedure ought to remove many bids, or whether the relationship between the average number of non-dominated bids and total bids ought to vary substantially from one distribution to another, so we set out to measure this relationship. Figure 7.4.1 shows the number of non-dominated bids as a function of the total number of bids generated. In these experiments, with each line representing an average over 20 runs, bids were generated for an auction with 64 goods, and the program stopped after 2000 non-dominated bids had been made. We observe that some of the legacy distributions are considerably more likely than others to generate non-dominated bids; we do not show the CATS distributions in this graph as all five generated virtually no dominated bids.

Of course, many other polynomial-time preprocessing steps are possible, *e.g.*, a check for bids that are dominated by a pair of other bids. Indeed, CPLEX employs

[10]An exception is that problems generally become easier when the number of bids grows *very* large in distributions favoring small bundles, because each small bundle is sampled much more often than each large bundle, giving rise to a new distribution for which the optimal allocation tends to involve only small bundles; *cf.*, [Anderson *et al.*, 2000].

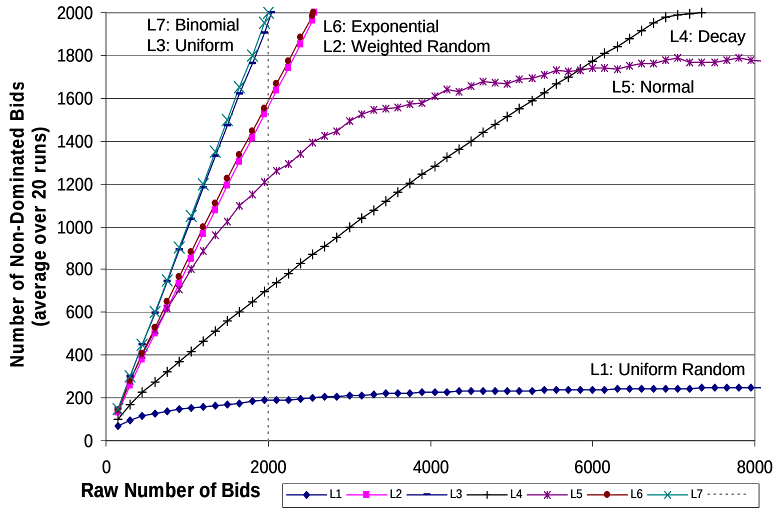


Figure 7.13: Non-Dominated Bids vs. Raw Bids

many, much more complex preprocessing steps before initiating its own branch-and-bound search. Our own experience with algorithms for the WDP has suggested that other polynomial-time preprocessing steps offer much poorer performance in terms of the number of bids discarded in a given amount of time. In any case, the results above suggest that strict domination checking should not be disregarded, since distributions differ substantially in the ratio between the number of non-dominated bids and the raw number of bids. The CATS software has the ability to generate instances for all CATS and legacy distributions with a specified number of *non-dominated* bids: the software iteratively generates bids and removes dominated bids until the specified target is reached. Observe that if we want to be able to generate any given number of non-dominated bids then we will be unable to use the distributions L1 and L5, because they often fail to generate a target number of non-dominated bids even after millions of bids were created. (This helps explain why L1 and L5 have been found empirically easy by other researchers.)

7.4.2 Sampling Parameters

In our original paper on CATS [Leyton-Brown *et al.*, 2000a], we suggested default values for the parameters of each generator. These defaults represented reasonable

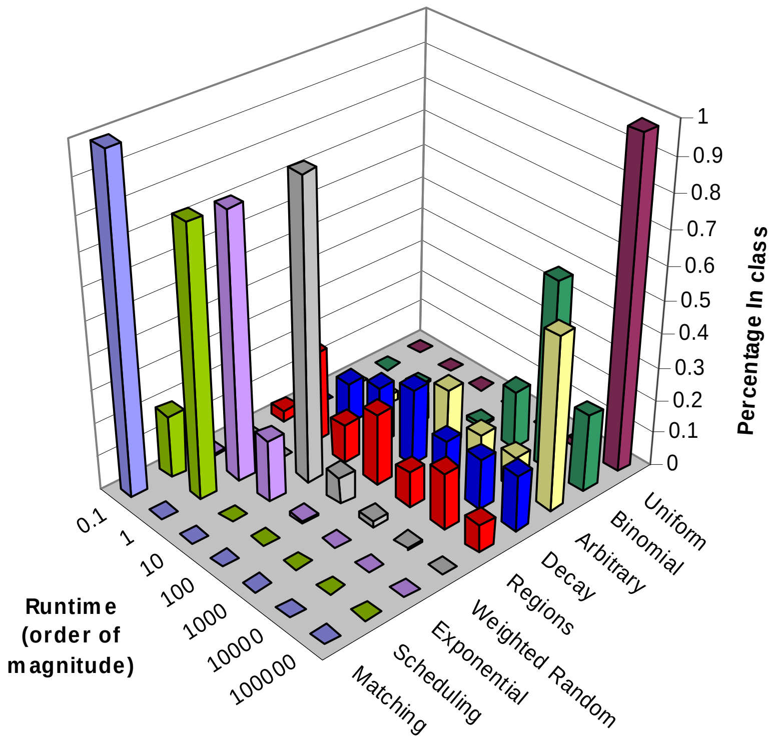


Figure 7.14: Gross Hardness

choices for the parameter values; however, the parameter space is large and the computational characteristics of the different CATS distributions may vary substantially throughout this space. An alternative to the use of default values for each parameter is the establishment of reasonable ranges for each parameter. Whenever an instance is created, a value for each parameter can be sampled uniformly at random from this range, ensuring that the whole parameter space is explored. Newer versions of the CATS software support this sort of parameter sampling.

7.4.3 Making CATS Harder

There has been discussion in the combinatorial auctions literature about whether CATS is computationally hard (see, *e.g.*, [Gonen & Lehmann, 2000; Sandholm *et al.*, 2001]). We performed tests on both CATS and legacy distributions with ILOG's CPLEX solver, sampling parameters as described above. Figure 7.14 shows the results of 500 runs for each distribution on problems with 256 goods and 1000 non-dominated bids, indicating the number of instances with the same order-of-magnitude runtime—*i.e.*, $\lfloor \log_{10}(runtime) \rfloor$. Each instance of each distribution had different parameters, each of which was sampled from a range of acceptable values.

We can see that several of the CATS distributions are quite easy for CPLEX, and that others vary from easy to hard. It is interesting that most distributions had instances that varied in hardness by several orders of magnitude, despite the fact that all instances had the same problem size. This gives rise to the question of whether we can tune CATS so that in addition to generating "realistic" instances, it also generates the hardest possible instances? We present techniques that answer this question in Section 10.4.1 in Chapter 10.

Our interest in generating the hardest possible instances notwithstanding, we should not be discouraged by the fact that some CATS distributions are computationally easy. On the contrary, this evidence suggests that realistic bidding patterns may often lead to much more tractable winner determination problems than the hardest unrealistic distributions such as "uniform". This is good news for those who hope to run practical combinatorial auctions.

7.5 Conclusions

In this chapter we introduced CATS, a test suite for combinatorial auction optimization algorithms. The distributions in CATS represent a step beyond previous CA testing techniques because they are economically motivated and model real-world problems. In the next chapter we use these distributions to evaluate CASS.

Chapter 8

Evaluating Combinatorial Auction Algorithms

This chapter presents experimental results for CASS.[1] First, scaling experiments demonstrate that CASS can scale exponentially in the number of goods, but appears to scale subexponentially in the number of bids. Second, CASS's anytime performance is examined: CASS can find nearly-optimal solutions orders of magnitude sooner than it terminates, and can spend considerable time proving optimality after finding the optimal solution. Third, CASS is contrasted with the Bidtree algorithm, another widely-cited WDP algorithm. Finally, CASS is contrasted with the latest version of ILOG's CPLEX software.

8.1 Original CASS Experiments

All the experiments in this section were run a 450MHz Pentium II with 256MB of RAM running Windows NT 4.0. CASS was implemented in ANSI C++, and is publicly available. While these experiments were run on older hardware, they are still useful for gaining an understanding of the algorithm. They examine all legacy

[1]We do not present experimental results for CAMUS here, since Chapter 7 focused on single-unit distributions, and Chapter 9 will go on to consider only the single-unit WDP. For an experimental evaluation of CAMUS, please see [Leyton-Brown *et al.*, 2000b].

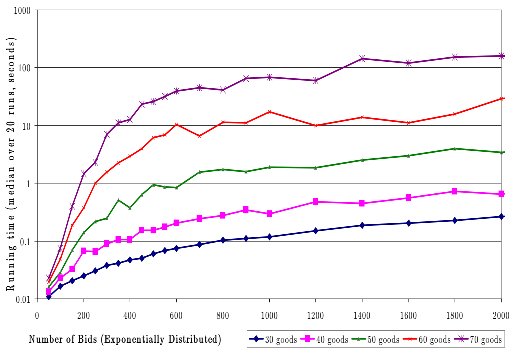


Figure 8.1: CASS Scaling: L6

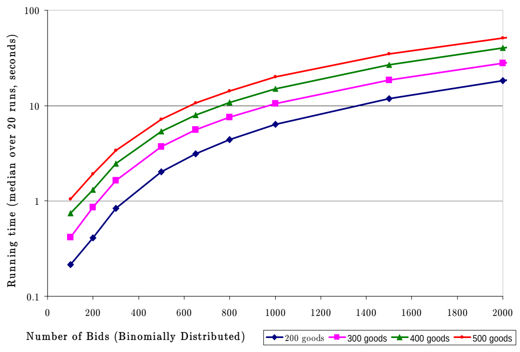


Figure 8.2: CASS Scaling: L7

distributions discussed in Section 7.3.7 except for L5 and L8.

8.1.1 Scaling Performance

The experiments in this section help us to understand CASS's performance as the number of bids[2] and goods is varied. Figures 8.1 and 8.2 show CASS's performance on the L6 and L7 distributions. Observe that for L6 runtime appears to scale exponentially in the number of goods (the four plots, representing linear increases in the number of goods, are spaced roughly equally on the log plot); for L7 runtime appears to scale sub-exponentially in the number of goods. For both L6 and L7 runtime clearly scales polynomially in the number of bids, as all the curves are sublinear on a linear axis.

8.1.2 Anytime Performance

Figure 8.3 shows CASS's anytime performance. Observe that the time it takes CASS to find the optimal solution is nearly of an order of magnitude smaller than the time at which it terminates: this is because *proving* optimality occupies most of the search. (Contrast this behavior with other optimization algorithms such as A*, which never finds the optimal solution before optimality is proven.) Also observe that the time it

[2]Unlike the data in Section 8.2 and in later chapters, the experiments in this section consider raw numbers of bids rather than numbers of undominated bids.

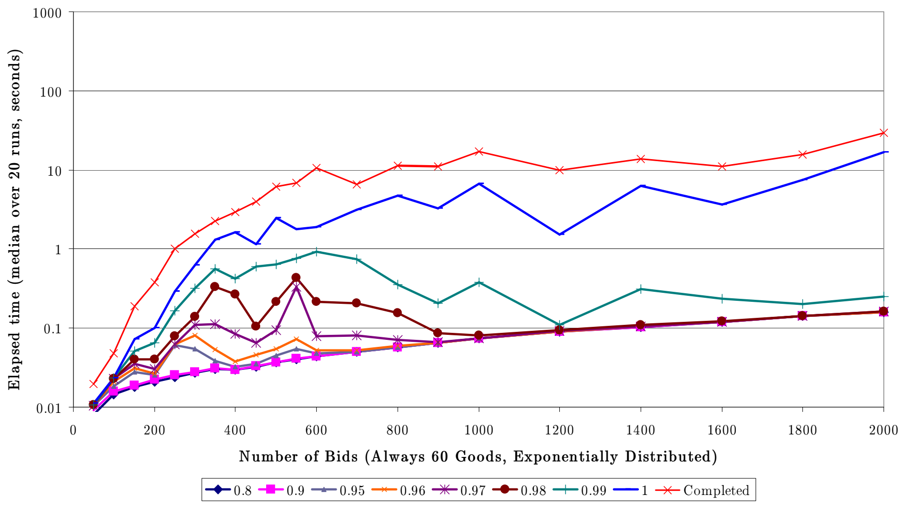


Figure 8.3: CASS Anytime Performance: L6

takes CASS to finds a solution within 99% of optimality is usually *much* smaller—as much as two orders of magnitude. This means that CASS can be very useful as an anytime algorithm even in situations where it does not terminate in a reasonable amount of time. Of course, however, there is no theoretical guarantee that CASS will always find a solution that is close to optimal.

8.1.3 CASS vs. Bidtree

Besides CASS, Bidtree is the other special-purpose WDP algorithm that has been most widely studied and cited in the literature. It was presented in the same conference proceedings as CASS [Sandholm, 1999]. The Bidtree algorithm is similar to CASS in several ways, but important differences hold. In particular, Bidtree performs a secondary depth-first search to identify non-conflicting bids, whereas CASS's structured approach provides context to the upper bound function as well as allowing it to avoid considering most conflicting bids. Bidtree also performs no caching or cache pruning. On the other hand, Bidtree uses an IDA* search strategy rather than CASS's branch-and-bound approach, and does more preprocessing.

Figures 8.4, 8.5, 8.6 and 8.7 contrast CASS and Bidtree's performance on the L1, L2, L3 and L4 distributions respectively. The Bidtree algorithm has never been made

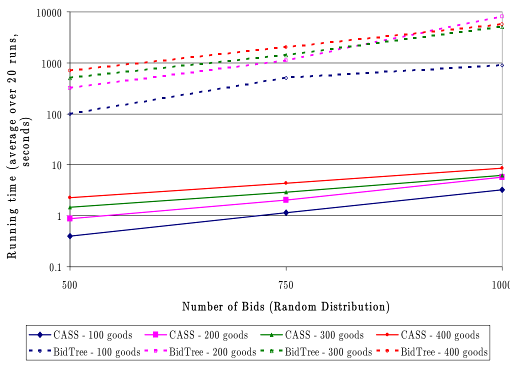


Figure 8.4: CASS vs. Bidtree: L1

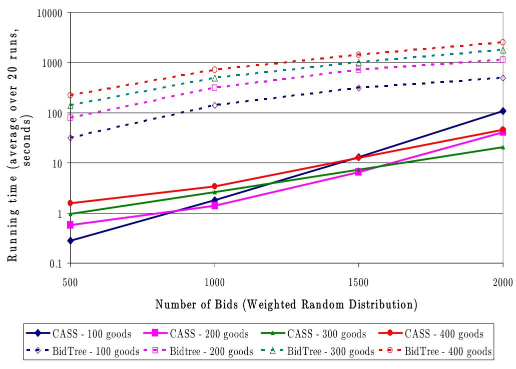


Figure 8.5: CASS vs. Bidtree: L2

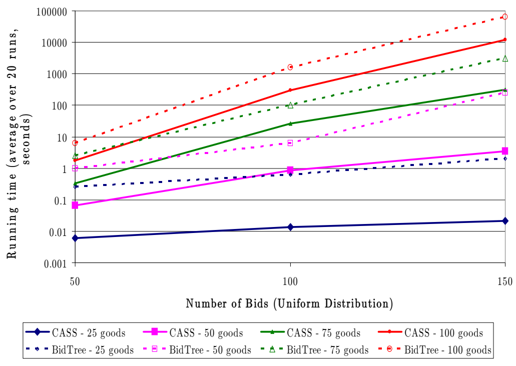


Figure 8.6: CASS vs. Bidtree: L3

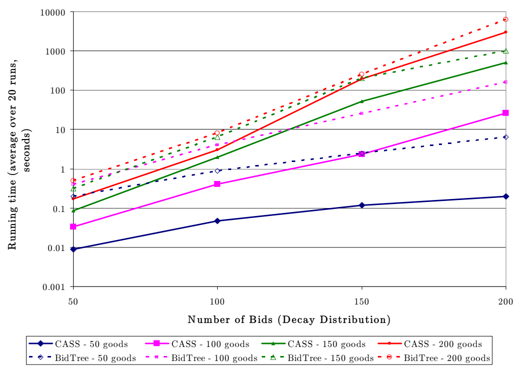


Figure 8.7: CASS vs. Bidtree: L4

publicly available; Bidtree performance for these figures was taken from [Sandholm, 1999]. It is therefore impossible to rerun these experiments on other distributions or varying the number of nondominated bids instead of the number of raw bids. Observe that overall, CASS dramatically outperforms Bidtree: CASS is between 2 and 500 times faster than Bidtree on the data points shown here, and is never slower.

8.2 CASS vs. CPLEX

When CASS and Bidtree were proposed, ILOG's CPLEX 5 mixed integer programming package (the industry standard) was unable to solve most WDP problems within a reasonable amount of time. Since that time, however, CPLEX's mixed integer programming module improved substantially with version 6 (released 2000), and substantially again with version 7 (released 2001). Now that CPLEX is in version 8, there has been general convergence in the research community towards using CPLEX as the default approach for solving the WDP.

The only ongoing effort at competition with CPLEX has come from the authors of Bidtree, who have written an updated algorithm called CABOB which they claim is much faster [Sandholm *et al.*, 2001]. However, like Bidtree, CABOB is not available to researchers. This is a serious impediment because published runtime data for CABOB is insufficient for our purposes: from this point forward we analyze algorithms in more detail than simply comparing average or median running times. In any case, CABOB's reported performance is similar to CPLEX's, and CABOB is also similar to CPLEX in its construction: it makes use of CPLEX's linear programming package as a subroutine and uses a similar search strategy. For these reasons, we present no experiments with CABOB.

In this section we compare the performance of CASS and CPLEX 8.0. These experiments were run on a cluster of 12 dual 2.4 Ghz Xeon machines with 1 GB RAM running Redhat Linux. We tested on 10 of the distributions provided by the CATS suite: all those distributions capable of generating an arbitrary number of nondominated bids. Specifically, we used all five of the CATS distributions (paths, regions, arbitrary, matching, scheduling) as well as five of the legacy distributions (L2, L3, L4, L6, L7). We sampled each of the distributions' parameters from a hand-chosen range of "reasonable" parameters as described in Section 7.4.2. We also sampled the problem size parameters: number of goods was chosen uniformly from $[40, 400]$ and number of nondominated bids was chosen uniformly from $[50, 2000]$. The full dataset had roughly 100 data points per distribution for a total of about 1000 data points, and took nearly 8 months of CPU time to collect. In order to increase the number of data points we were able to collect during this time, we capped CASS's runtime at 12 hours.

Figure 8.8 shows the mean runtime of CASS and CPLEX on each distribution. Note that the vertical axis uses a logarithmic scale so that all the bars can be seen on

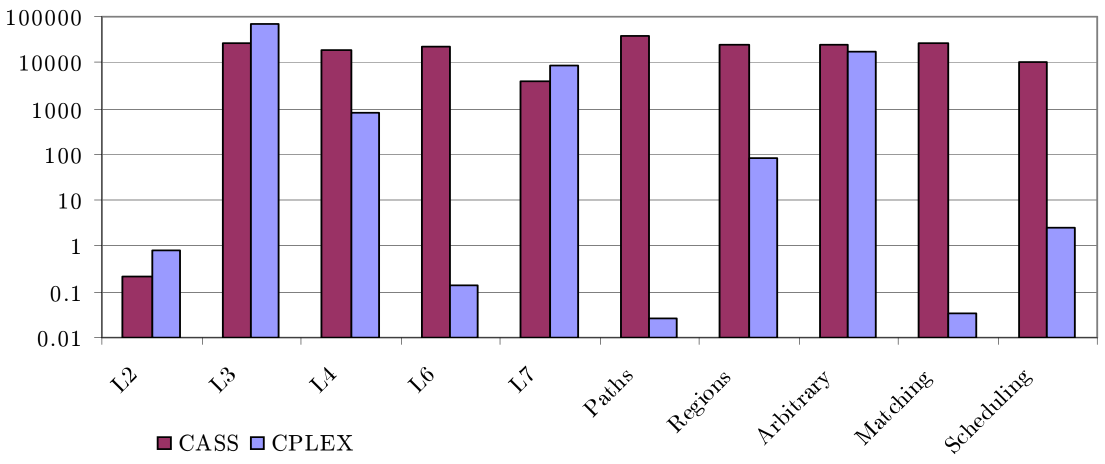


Figure 8.8: CASS vs. CPLEX: mean runtime per distribution

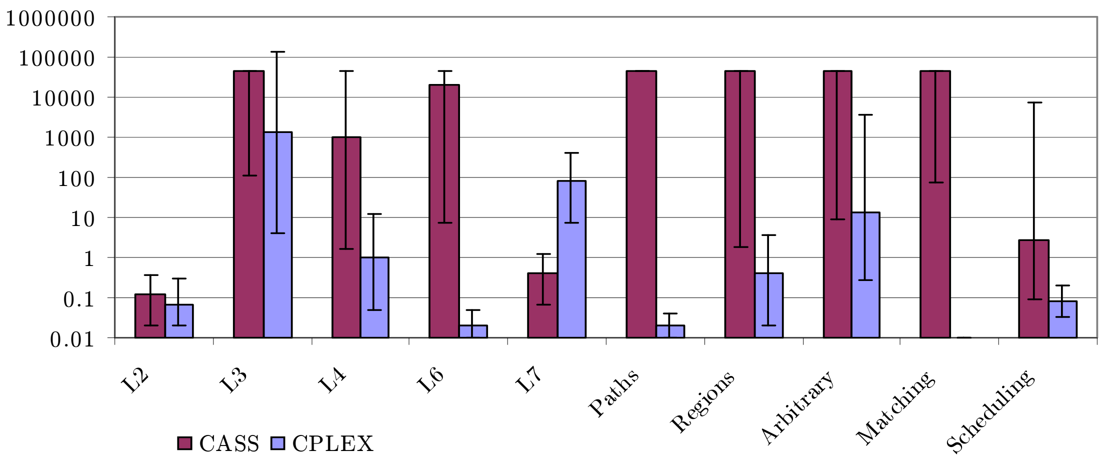


Figure 8.9: CASS vs. CPLEX: first, second and third quartiles

the same graph. Judging from this picture CPLEX seems to be a better choice than CASS most of the time. Exceptions are L3 and L7, where CASS does much better than CPLEX, and L2 where the difference is less pronounced.

Average performance can be overwhelmed by a relatively small fraction of runs that take a very long time. To learn more about how CASS and CPLEX 8.0 compare, we can graph medians instead of means. Figure 8.9 shows medians for each distribution; the error bars indicate first and third quartiles (of course, median is the second quartile). We can see several things from this graph. First, on the whole CPLEX still appears to be much faster than CASS. Note L3: while CASS had better average performance on this distribution, CPLEX has better median performance. On L7 CASS's advantage is now shown to be much larger, with the two algorithms' error bars not even overlapping. On this distribution it does seem that a small number of very hard instances skewed CASS's average upwards; the same is true for CASS on L4 and scheduling, and for CPLEX on L4, L7, regions and arbitrary. Overall, we can see from the error bars that most distributions exhibit substantial runtime variation.

In Figure 8.9 it is also the case that the error bars often overlap. This raises the question of the extent to which the algorithms' runtimes are uncorrelated. If this level is high then there could be great benefit to running both CASS and CPLEX in parallel (or in choosing between the algorithms in a more sophisticated way: see Chapter 10). To investigate the level of correlation between CASS and CPLEX on a per-instance basis, we plotted the algorithms' runtimes on separate axes of a scatter plot in Figure 8.10. For 5% of the instances, CASS and CPLEX had the same running time within two significant digits, and on 6% of the instances both CASS and CPLEX took longer than CASS's cap time of 12 hours, making it impossible to compare the algorithms. As we expected given its much better running times in Figures 8.8 and 8.9, CPLEX outperformed CASS a large fraction of the time (67%). The surprise is that there remained a substantial fraction (22%) of instances on which CASS outperforms CPLEX; on many instances the performance difference was very significant.

8.3 Conclusions

This chapter detailed experimental investigations of the CASS algorithm's performance. First, it was shown to have good anytime performance, finding good solutions almost immediately and finding an optimal solution long before optimality is proven. Second, it was compared to Sandholm's Bidtree algorithm, which it consistently and significantly outperformed. Finally, it was compared to a more modern algorithm, ILOG's CPLEX 8.0. On most (but not all) test distributions, CPLEX exhibited considerably better performance. However, a more careful analysis showed

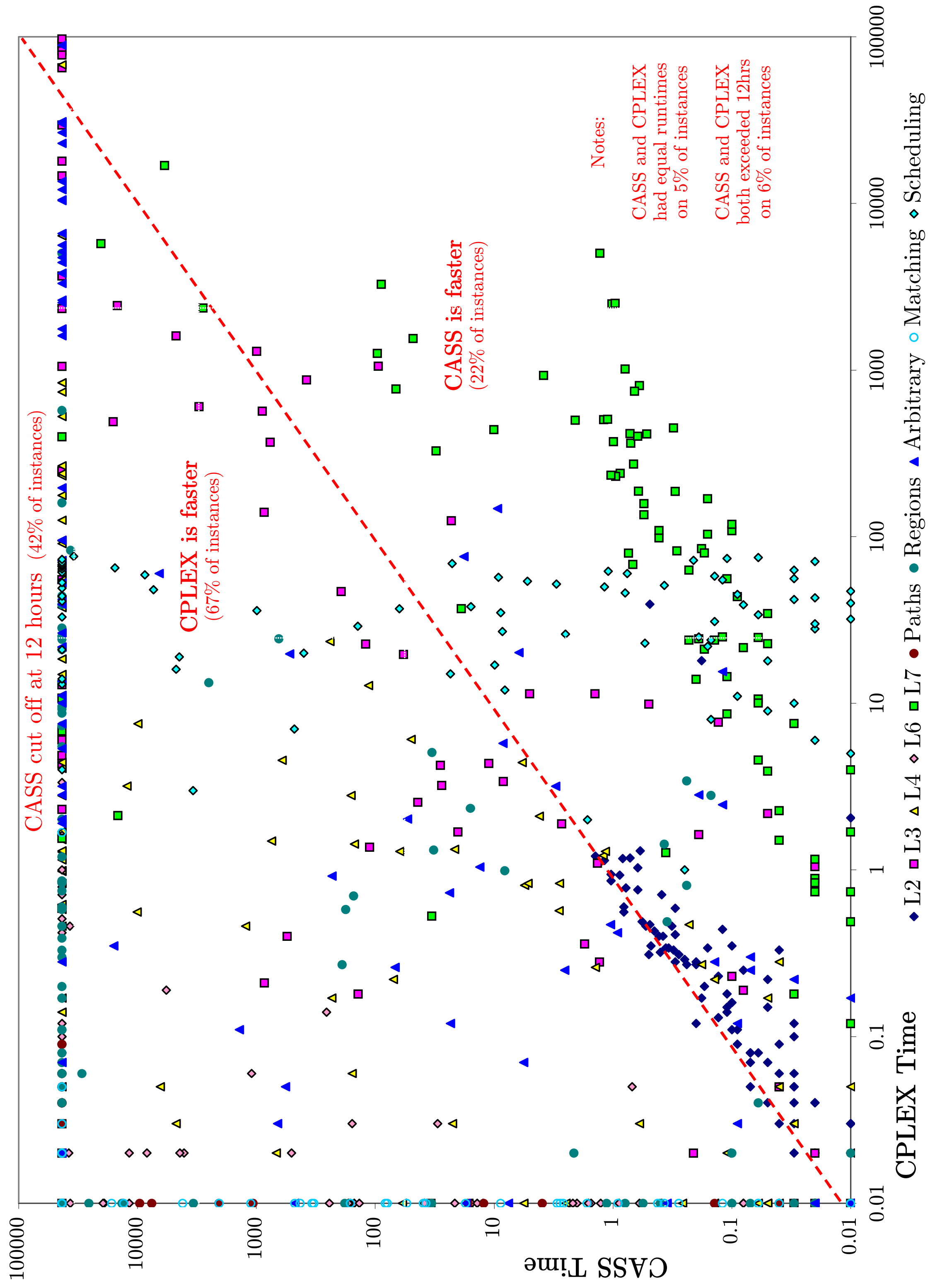


Figure 8.10: CASS vs. CPLEX: Scatter Plot

that the two algorithms' performance was often uncorrelated, and that there were a substantial fraction of these instances on which CASS dramatically outperformed CPLEX. In Chapter 10 we will revisit this uncorrelation between CASS and CPLEX and explore ways of leveraging it to build a better WDP algorithm. First, however, we will study ways of explaining why an algorithm (such as CPLEX) shows so much runtime variation on similar problems.

Chapter 9

Empirical Hardness Models

We propose a new approach for understanding the algorithm-specific empirical hardness of $\mathcal{NP}$-hard problems. We use machine learning models to build regression models that predict an algorithm's runtime given a novel problem instance. We also discuss techniques for interpreting these models to gain understanding of the characteristics that cause instances to be hard or easy.

This chapter maintains our focus on the WDP, describing our attempts to build models to predict CPLEX's performance on nine[1] of the CATS distributions. However, we also attempt to emphasize the generality of our methodology in order to encourage its application to other $\mathcal{NP}$-hard problems.

9.1 Empirical Hardness

It is often the case that particular instances of $\mathcal{NP}$-hard problems can be quite easy to solve in practice. In recent years researchers in the constraint programming and artificial intelligence communities have studied the *empirical* hardness of individual instances or distributions of $\mathcal{NP}$-hard problems, and have often managed to find simple mathematical relationships between features of the problem instances and the hardness of the problem. The majority of this work has focused on decision problems:

[1]The paths distribution is missing from this chapter and the next because of a technical glitch. It will appear in the final version of this thesis.

that is, problems that ask a yes/no question of the form, “Does there exist a solution meeting the given constraints?”. The most successful approach for understanding the empirical hardness of such problems—taken for example in [Cheeseman *et al.*, 1991; Achlioptas *et al.*, 2000]—is to vary some parameter of the input looking for a hard-easy-hard transition corresponding to a phase transition in the solvability of the problem. This approach uncovered the famous result that 3-SAT instances are hardest when the ratio of clauses to variables is about 4.3; it has also been applied to other decision problems such as quasigroup completion [Gomes & Selman, 1997]. Another approach rests on a notion of backbone [Monasson *et al.*, 1998; Achlioptas *et al.*, 2000], which is the set of solution invariants.

9.1.1 Optimization Problems

Some researchers have also examined the empirical hardness of optimization problems, which ask a real-numbered question of the form, “What is the best solution meeting the given constraints?”. These problems are clearly different from decision problems, since they always have solutions and hence cannot give rise to phase transitions in solvability. One way of finding hardness transitions related to optimization problems is to transform them into decision problems of the form, “Does there exist a solution with the value of the objective function $\geq x$?” This approach has yielded promising results when applied to MAX-SAT and TSP. Other experimentally-oriented work includes the extension of the concept of backbone to optimization problems [Slaney & Walsh, 2001], although it is often difficult to define for arbitrary problems and can be costly to compute.

A second approach is to attack the problem analytically rather than experimentally. For example, Zhang [1999] performed average case theoretical analysis of particular classes of search algorithms. Though his results rely on independence assumptions about the branching factor and heuristic performance at each node of the search tree that do not generally hold, the approach has made theoretical contributions—describing a polynomial/exponential-time transition in average-case complexity—and shed light on real-world problems. Korf and Reid [1998] predict the average number of nodes expanded by a simple heuristic search algorithm such as A* on a particular problem class by making use of the distribution of heuristic values in the problem space. As above, strong assumptions are required: *e.g.*, that the branching factor is constant and node-independent, and that edge costs are uniform throughout the tree.

Both experimental and theoretical approaches have sets of problems to which they are not well suited. Existing experimental techniques have trouble when problems have high-dimensional parameter spaces, as it is impractical to manually explore the space of all relations between parameters in search of a phase transition or some other predictor of an instance's hardness. This trouble is compounded when many different data distributions exist for a problem, each with its own set of parameters. Theoretical approaches are also difficult when the input distribution is complex or is otherwise hard to characterize, but they also have other weaknesses. They tend to become intractable when applied to complex algorithms, or to problems with variable and interdependent edge costs and branching factors. Furthermore, they are generally unsuited to making predictions about the empirical hardness of individual problem instances, instead concentrating on average (or worst-case) performance on a class of instances.

9.1.2 The WDP

The WDP is a good example of a problem that is ill-suited to study by either existing experimental or theoretical approaches: instances are characterized by a large number of apparently relevant features, many, highly parameterized distributions exist, there is significant variation in edge costs throughout the search tree and it is desirable to predict the empirical hardness of individual problem instances. It has often been observed that WDP algorithms vary by many orders of magnitude in their running times for different problems of the same size—even for different instances drawn from the same distribution. However, little is known about what causes WDP instances to vary so substantially in their empirical hardness. Understanding what characteristics of data instances are predictive of long running times would be useful for predicting how long an auction will take to clear, tuning data distributions for hardness, constructing algorithm portfolios, designing package bidding rules to reduce the chances of long clearing times and possibly for improving the design of WDP algorithms.

9.2 Building Hardness Models

9.2.1 Our Methodology

We propose a novel experimental approach for predicting the running time of a given algorithm on individual instances of such a problem, drawn from one of many different distributions. Our methodology follows:

1. An optimization algorithm is selected.

2. A set of problem instance distributions is selected. For each parameter of each distribution a range of acceptable values is established.

3. Problem size is defined and a size is chosen. Problem size will be held constant to focus on unknown sources of hardness.

4. A set of polytime-computable, distribution-independent features is selected.

5. To generate instances, a distribution is chosen at random and then the range of acceptable values for each parameter is sampled. This step is repeated until the desired number of problem instances have been generated.

6. For each problem instance the running time of the optimization algorithm is determined, and all features are computed.

7. Redundant or uninformative features are eliminated.

8. A function of the features is learned to predict running time (or some other measure of empirical hardness), and prediction error is analyzed.

9.2.2 Problem Size

Some sources of empirical hardness in $\mathcal{NP}$-hard problem instances are well understood. Our goal in this chapter is to understand what *other* features of instances are predictive of hardness so we hold these parameters constant, concentrating on variations in other features.

For the WDP, it is known that instances generally become harder as the problem gets larger: *i.e.*, as the number of bids and goods increases. Our goal is to understand what *other* features of instances are predictive of hardness so we hold these parameters constant, concentrating on variations in other features. As argued in Chapter 7, the removal of dominated bids can have a significant effect. We therefore defined problem size as the pair (*number of goods, number of non-dominated bids*).

9.2.3 Features

In order to learn a model we must first characterize each problem instance with a set of features, obtained using domain knowledge to identify properties of the instance that appear likely to provide useful information about empirical hardness. We only consider features that can be generated from *any* problem instance, without knowledge of how that instance was constructed. Furthermore we restrict ourselves to those features that are computable in polynomial time, since the computation of the features should scale well as compared to solving the optimization problem. Although features must be manually selected, there are statistical techniques for identifying useless features. Identifying such features is important because highly correlated features can unnecessarily increase the dimensionality of the hypothesis space: this can degrade the performance of some regression algorithms and also makes the resulting formula harder to interpret.

For our WDP case study, we determined 35 features which we thought could be relevant to the empirical hardness of the optimization, ranging in their computational complexity from linear to cubic time. After feature selection we were left with 25 features: these are summarized in Figure 9.1. We describe our features in more detail below, and also mention some of the features that we dropped.

Bid-Good Graph Features:

1-3. **Bid nodes degree statistics**: max and min degree of the bid nodes, and standard deviations.

4-7. **Good nodes degree statistics**: average, maximum, minimum degree of the good nodes, and their standard deviations.

Bid Graph Features:

8. **Edge Density**: number of edges in the BG divided by the number of edges in a complete graph with the same number of nodes.

9-11. **Node degree statistics**: the max and min node degrees in the BG, and their standard deviation.

12-13. **Clustering Coefficient and Deviation**. A measure of "local cliqueness." For each node calculate the number of edges among its neighbors divided by $k(k-1)/2$, where k is the number of neighbors. We record average (the clustering coefficient) and standard deviation.

14. **Average minimum path length**: the average minimum path length, over all pairs of bids.

15. **Ratio of the clustering coefficient to the average minimum path length**: One of the measures of the smallness of the BG.

16-19. **Node eccentricity statistics**: The eccentricity of a node is the length of a shortest path to a node furthest from it. We calculate the maximum eccentricity of BG (graph diameter), the minimum eccentricity of BG (graph radius), average eccentricity, and standard deviation of eccentricity.

LP-Based Features:

20-22. ℓ_1, ℓ_2, ℓ_∞ norms of the integer slack vector.

Price-Based Features:

23. **Standard deviation of prices among all bids**: $stdev(p_i)$

24. **Deviation of price per number of goods**: $stdev(p_i/|b_i|)$

25. **Deviation of price per square root of the number of goods**: $stdev(p_i/\sqrt{|b_i|})$.

Figure 9.1: Four Groups of Features for the WDP

There are two natural graphs associated with each instance. First, is the *bid-good graph* (BGG): a bipartite graph having a node for each bid, a node for each good and an edge between a bid and a good node for each good in the given bid. We measure a variety of BGG's properties: extremal and average degrees and their standard deviations for each group of nodes. The average number of goods per bid was perfectly correlated with another feature, and so did not survive our feature selection.

The *bid graph* (BG) represents conflicts among bids (thus it is the constraint graph for the associated CSP). As is true for all CSPs, the BG captures a lot of useful information about the problem instance. Our second group of features are concerned with structural properties of the BG. We originally measured the first and third quartiles and the median of the BG node degrees, but they turned out to be highly correlated with edge density. We also measured the average number of conflicts per bid, but since the number of bids was held constant this feature was always proportional to edge density. We considered using the number of connected components of the BG to measure whether the problem is decomposable into simpler instances, but found that nearly every instance consisted of a single component. Finally, it would be desirable to include some measure of the size of the (unpruned) search space. For some problems branching factor and search depth are used; for WDP neither is easily estimated. A related measure is the number of maximal independent sets of BG, which corresponds to the number of feasible solutions. However, this counting problem is hard, and to our knowledge does not have a polynomial-time approximation.

The third group of features is calculated from the solution vector of the LP relaxation linear programming relaxation of the WDP. We calculate the *integer slack vector* by replacing each component x_i with $min(|x_i|, |1 - x_i|)$. These features appeared particularly useful both because the slack gives insight into the quality of CPLEX's initial solution and because CPLEX uses LP as its search heuristic. Originally we also included median integer slack, but excluded the feature when we found that it was always zero.

Our last group of features is the only one to explicitly consider the prices associated with bids. While the scale of the prices has no effect on hardness the spread is crucial, since it impacts pruning. We note that feature 25 was shown to be an optimal bid-ordering heuristic for branch-and-bound search on the WDP in [Gonen & Lehmann, 2000].

9.3 Evaluating Hardness Models

We generated three separate data sets of different problem sizes, to ensure that our results were not artifacts of one particular choice of problem size. The first data set contained runs on instances of 1000 bids and 256 goods each, with a total of 4500 instances (500 instances per distribution). The second data set with 1000 bids and 144 goods had a total of 2080 instances; the third data set with 2000 bids and 64 goods contained 1964 instances. Where we present results for only a single data set, the first data set was always used. All of our runtime data was collected by running CPLEX 7.1 with preprocessing turned off.[2] We used a cluster of 4 machines, each of which had 8 Pentium III Xeon 550 MHz processors and 4G RAM and was running Linux 2.2.12. Since many of the instances turned out to be exceptionally hard, we stopped CPLEX after it had expanded 130,000 nodes (reaching this point took between 2 hours and 22 hours, averaging 9 hours). Overall, solution times varied from as little as 0.01 seconds to as much as 22 hours. We estimate that we consumed approximately 3 years of CPU time collecting this data. We also computed our 35 features for each instance. (Recall that feature selection took place after all instances had been generated.) Each feature in each data set was normalized to have a mean of 0 and a standard deviation of 1. Regression was performed using the open-source R package (see `www.r-project.org`).

Since we wanted to learn a continuous-valued model of the features, we used statistical regression techniques.[3] We chose the logarithm of CPLEX running time as our response variable—the value to be predicted—rather than absolute running time, because we wanted the model to be penalized according to whether the predicted and actual values had the same order of magnitude. If we had tried to predict absolute running times then the model would have been penalized very little for dramatically mispredicting the running time of very easy instances, and would have been penalized heavily for slightly mispredicting the running time of the hardest instances. We performed regression on a training set consisting of 80% of a dataset, and then tested our model on the remaining 20% to evaluate its ability to generalize to unseen data.

9.3.1 Linear Models

One of the simplest and most widely-studied regression techniques is linear regression. This technique finds a hyperplane in the feature space that minimizes root mean squared error (RMSE), the square root of the average squared difference between

[2]When the work described in this chapter was performed, CPLEX 7.1 was the latest version. Unfortunately, it's not easy to rerun 3 CPU-years worth of experiments! On the bright side, limited experiments suggest that CPLEX 8.0 is not a huge improvement over CPLEX 7.1, at least for our WDP benchmark distributions.

[3]A large literature addresses the statistical techniques we used; for an introduction see, *e.g.*, [Hastie *et al.*, 2001].

Data point	**Mean Abs Err**	**RMSE**	**Adj-R^2**
1000 Bids/256 Goods	0.399	0.543	0.938
1000 Bids/144 Goods	0.437	0.579	0.909
2000 Bids/64 Goods	0.254	0.368	0.912

Table 9.1: Linear Regression: Errors and Adjusted R^2

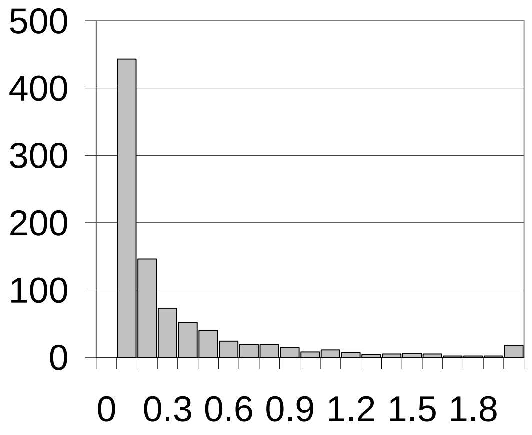


Figure 9.2: Linear Regression: Squared Error (test data)

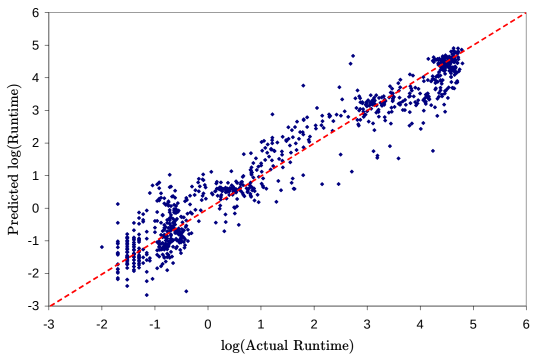


Figure 9.3: Linear Regression: Absolute Error Scatterplot (test data)

the predicted value and the true value of the response variable. Minimizing RMSE is reasonable because it conforms to the intuition that, holding mean absolute error constant, models that mispredict all instances equally should be preferred to models that vary in their mispredictions. Although we go on to consider nonlinear regression, it is useful to consider the results of linear regression for two reasons. First, one of our main goals was to understand the factors that influence hardness, and insights gained from a linear model are useful even if other, more accurate models can be found. Second, our linear regression model serves as a baseline to which we can compare the performance of more complex regression techniques.

In Figure 9.1 we report both RMSE and mean absolute error, since the latter is often more intuitive. A third measure, adjusted R^2, is the fraction of the original variance in the response variable that is explained by the model, with a penalty for more complex models. Adjusted R^2 is a measure of fit to the training set and cannot entirely correct for overfitting; nevertheless, it can be an informative measure when

Data point	**Mean Abs. Err.**	**RMSE**	R^2
1000 Bids/256 Goods	0.183	0.297	0.987
1000 Bids/144 Goods	0.272	0.475	0.974
2000 Bids/64 Goods	0.163	0.272	0.981

Table 9.2: Quadratic Regression: Errors and Adjusted R^2

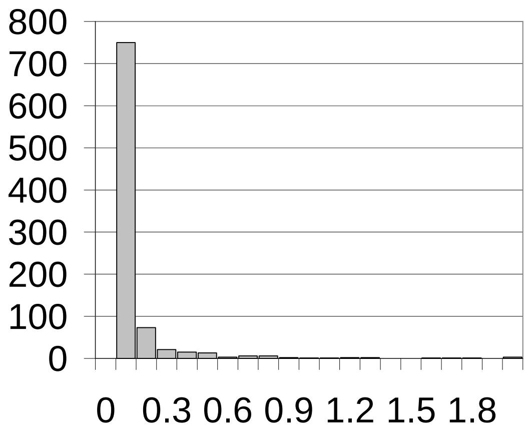


Figure 9.4: Quadratic Regression: Squared Error (test data)

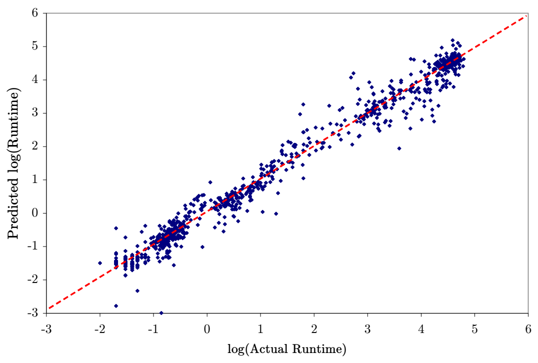


Figure 9.5: Quadratic Regression: Error Scatterplot (test data)

presented along with test set error. Figure 9.2 shows a histogram of the RMS error, with bin width 0.1. Figure 9.3 shows a scatterplot of predicted $\log_{10}$ runtime vs. actual $\log_{10}$ runtime. We can see from these two figures that most instances are predicted very accurately, and few instances are dramatically mispredicted. Overall, these results show that our linear model would be able to do a good job of classifying instances into the bins shown in Figure 7.14 in Section 7.4.3, despite the fact that it is not given the distribution from which each instance was drawn: 93% of the time the log running times of the data instances in our test set were predicted to the correct order of magnitude (*i.e.*, with an absolute error of less than 1.0).

9.3.2 Nonlinear Models

Although our linear model was quite effective, we expected that nonlinear interactions between our features would be important, and therefore we investigated nonlinear models. A simple way of performing nonlinear regression is to compute new features based on nonlinear interactions between the original features, and then to perform linear regression on the union of both sets of features. We added all products of pairs of features to our linear model, including squares of individual features, which gave us a total of 350 features. This allowed us to fit a 2^{nd} degree polynomial instead of our previous linear model. For all three of our datasets this model gave considerably better error measurements on the test set and also explained nearly all the variance in the training set, as shown in Table 9.2. As above, Figures 9.4 and 9.5 show a histogram of root mean squared error and a scatterplot of predicted log runtime vs. actual log runtime. Comparing these figures to Figures 9.2 and 9.3 confirms our judgment that the quadratic model is substantially better overall.

We also explored another nonlinear regression technique, Multivariate Adaptive Regression Splines (MARS) [Friedman, 1991]. MARS models are linear combinations of the products of one or more basis functions, where basis functions are the positive parts of linear functions of single features. The RMSE on our MARS models differed from the RMSE on our second-order model only in the second decimal place; as MARS models can be unstable and difficult to interpret, we focus on our second-order model.

9.4 Analyzing Hardness Models

The results summarized above demonstrate that it is possible to learn a model of our features that accurately predicts the log of CPLEX running time on novel instances. For some applications (*e.g.*, predicting the time it will take for an auction to clear; building an algorithm portfolio) accurate prediction is all that is required. For other applications it is necessary to *understand* what makes an instance empirically hard. In this section we set out to interpret our models.

9.4.1 Cost of Omission

It is tempting to interpret a model by comparing the coefficients assigned to the different features; since all features have the same mean and standard deviations, more important features should tend to have larger coefficients. The reason that this does not work is that most of our features are at least somewhat correlated. Two perfectly correlated but entirely unimportant features can have large coefficients with opposite signs in a linear model; in practice, since imperfect correlation and correlations among larger sets of variables are common, it is difficult to untangle the effects of correlation and importance in explaining a given coefficient's magnitude. One solution is to force the model to have smaller coefficients and/or to contain fewer variables. Requiring smaller coefficients reduces interactions between correlated variables; two popular techniques are ridge regression and lasso regression. We evaluated these techniques using cross validation and found no significant effect on errors or on interpretability of the model, so we do not present these results here. Likewise, single-variable models were not informative on our data: the best such model had an RMSE error of nearly 2 in predicting $\log_{10}$ running time.

Another family of techniques allows interpretation without the consideration of coefficient magnitudes. These techniques select "good" subsets of the features, essentially performing exhaustive enumeration when possible and various greedy search techniques otherwise. Small models are desirable for analysis because they are easier to interpret directly and because a small, optimal subset will tend to contain fewer highly covariant features than a larger model. (For a detailed discussion of techniques for selecting relevant feature subsets and for comparisons of different definitions of "relevant," focusing on classification problems, see [Kohavi & John, 1997].)

We plotted subset size (from 1 to the total number of variables) versus the RMSE of the best model built from a subset of each size. We then chose the smallest subset size at which there was little incremental benefit gained by moving to the next larger subset size. We examined the features in the model, and also measured each variable's cost of omission—the (normalized) difference between the RMSE of the model on the original subset and a model omitting the given variable. It is important to note that because of correlation between features many different subsets may achieve nearly the same RMSE, and that very little can be inferred from what *particular* variables are included in the best subset of a given size. This is of little concern, however, when subset selection is used only to gain a conceptual understanding of the features that are important for predicting empirical hardness; in this case the substitution of

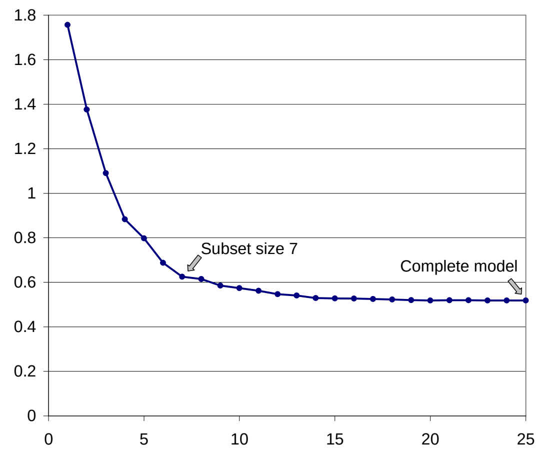


Figure 9.6: Linear Regression: Subset size vs. RMSE.

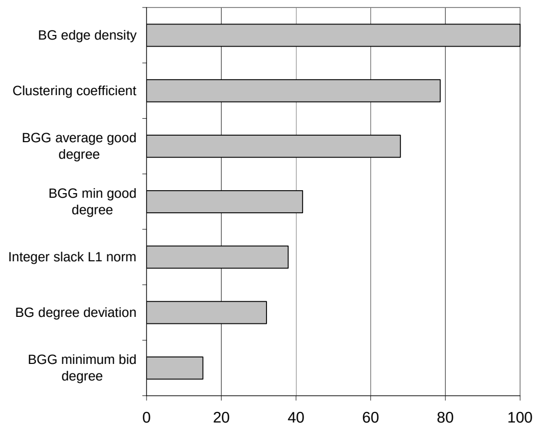


Figure 9.7: Linear Regression: Cost of omission for subset size 7.

one feature for another covariant feature is irrelevant because the inclusion of either feature in the model should have the same intuitive meaning. It is also worth noting that subset selection and cost of omission were both evaluated using the test set, but that all model selection was evaluated using cross validation, and all analysis was performed after our models had been learned.

9.4.2 Experimental Results

Figure 9.6 shows the RMSE of the best subset containing between 1 and 25 features for linear models; since we had only 25 features in total we selected the best subsets by exhaustive comparison. We chose to examine the model with seven features because it was the first for which adding another feature did not cause a large decrease in RMSE. Figure 9.7 shows the seven features in this model and their respective costs of omission (scaled to 100). The most striking conclusion is that structural features are the most important. Edge density of BG is essentially a measure of the constrainedness of the problem, so it is not surprising to find that this feature is the most costly to omit. Clustering coefficient, the second feature, is a measure of average cliquiness of BG; this feature gives an indication of how local the problem's constraints are. All but one of the remaining features concern node degrees in BG or BGG; the final feature

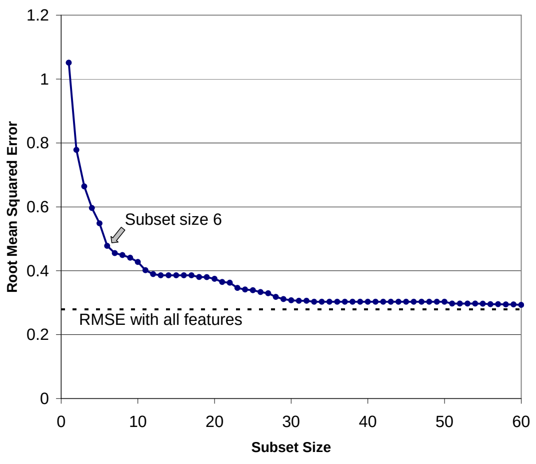


Figure 9.8: Quadratic Regression: Subset size vs. RMSE.

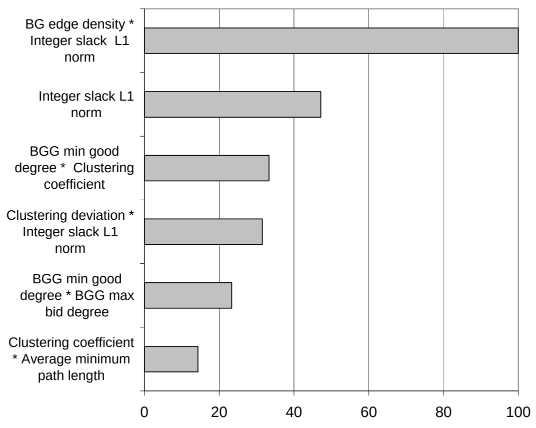


Figure 9.9: Quadratic Regression: Cost of omission for subset size 6.

is the ℓ_1 norm of the linear programming slack vector.

Figure 9.8 shows the best subsets containing between 1 and 60 features for second-order models. In this case we had 350 features, making exhaustive exploration of feature subsets impossible. Instead, we used three different greedy subset selection methods (forward selection; backward selection; sequential replacement) and at each size chose the best subset among the three. The subsets shown in Figure 9.8 are likely not the RMSE-minimizing subsets of the given sizes, but since our goal was only to understand what sorts of features are important this likely lack of optimality is not a serious problem. We observe that allowing interactions between features dramatically improved RMSE on very small subsets: our 5-feature quadratic model outperformed our 25-feature linear model.

Figure 9.9 shows the costs of omission for the variables from the best six-feature subset. As for our linear model, we observe that most critical features are structural: edge density of BG, the clustering coefficient and node degrees. Overall many second-order features were selected. The ℓ_1 norm becomes more important than in the linear model when it is allowed to interact with other features; in the second-order model it is also sufficiently important to be kept as the only first-order feature.

We can look at the features that were important to our quadratic and linear models in order to gain understanding about how our models work. The importance of the ℓ_1 norm is quite intuitive: the easiest problems can be completely solved by LP, yielding an ℓ_1 norm of 0; the norm is close to 0 for problems that are almost completely solved by LP (and hence often require less search to resolve), and larger for more difficult problems. The BG edge density feature describes the overall constrainedness of the problem. Generally, we would expect that very highly constrained problems would be easy, since more constraints imply a smaller search space; however, our experimental results show that CPLEX takes a long time on such problems. It seems that CPLEX's calculation of the LP bound at each node becomes much more expensive when the number of constraints in the LP increases substantially, and this cost overwhelms the savings that come from searching in a smaller space. Some other important features are intuitively similar to BG edge density. For example, the node degree statistics describe the max, min, average and standard deviation of the number of constraints in which each variable is involved; they indicate how quickly the search space can be expected to narrow as variables are given values (*i.e.*, as bids are assigned to the allocation). Similarly, the clustering coefficient features measure the extent to which variables that conflict with a given variable also conflict with each other, another indication as to the speed with which the search space will narrow as variables are assigned. Finally, we can now understand the importance of the feature which was by far the most important in our 6-feature quadratic model, the product of the BG edge density and the integer slack ℓ_1 norm. Note that this feature takes a large value only when both BG edge density and ℓ_1 norm are large; the explanations above show that problems are easy for CPLEX whenever either of these features has a small value. Since BG edge density and ℓ_1 norm are relatively uncorrelated on our data, their product gives a powerful prediction of an instance's hardness.

It is also interesting to notice which features were consistently *excluded* by subset selection. In particular, it is striking that no price features were important in either our first- or second-order models (except implicitly, as part of LP relaxation features). Although price-based features do appear in larger models, they seem not to be as critically important as structural or LP-based features. This may be partially explained by the fact that the removal of dominated bids eliminates the bids that deviate most substantially on price, and indeed caused us to eliminate the distribution (L1) in which average price per good varied most dramatically across bids. Another group of features that were generally not chosen for small subsets were path length features: graph radius, diameter, average minimum path length, etc. It seems that statistics derived from neighbor relations in constraint graphs are much more meaningful for predicting hardness than other graph-theoretic statistics derived from notions of proximity or connectedness.

9.5 Conclusions

We performed an extensive experimental investigation into the empirical hardness of the WDP. We identified structural, distribution-independent features of WDP instances and showed that, somewhat surprisingly, they contain enough information to predict CPLEX running time with high accuracy.

More importantly, we proposed a new, general methodology for understanding the empirical hardness of complex, high-dimensional $\mathcal{NP}$-hard problems. We believe that our methodology, based on using machine learning techniques to identify hard regions of the feature space, is applicable to a wide variety of hard problems. In the next chapter we will discuss ways of applying these models to build algorithm portfolios and to tune benchmark distributions for hardness.

Chapter 10

Applications of Hardness Models

This chapter describes techniques for building an algorithm portfolio that can outperform its constituent algorithms, and a method for generating test distributions to focus future algorithm design work on problems that are hard for an existing portfolio. We demonstrate the effectiveness of our techniques on the WDP. First, we show that a portfolio of CASS, CPLEX and GL [Gonen & Lehmann, 2001] outperforms CPLEX alone by a factor of three; second, we show that we are able to tune the CATS distributions to make them much harder for our portfolio.

10.1 Introduction

Although some algorithms are better than others on average, there is rarely a best algorithm for a given problem. Instead, it is often the case that different algorithms perform well on different problem instances. Not surprisingly, this phenomenon is most pronounced among algorithms for solving $\mathcal{NP}$-hard problems, because runtimes for these algorithms are often highly variable from instance to instance. When algorithms exhibit high runtime variance, one is faced with the problem of deciding which algorithm to use; Rice dubbed this the "algorithm selection problem" [Rice, 1976]. In the nearly three decades that have followed, the issue of algorithm selection has failed to receive widespread study, though of course some excellent work does exist. By far, the most common approach to algorithm selection has been to measure different algorithms' performance on a given problem distribution, and then to use only the algorithm having the lowest average runtime. This approach, to which we refer as "winner-take-all," has driven recent advances in algorithm design and refinement, but has resulted in the neglect of many algorithms that, while uncompetitive on average, offer excellent performance on particular problem instances. Our consideration of the algorithm selection literature, and our dissatisfaction with the winner-take-all approach, has led us to ask the following two questions. First, what general techniques can we use to perform per-instance (rather than per-distribution) algorithm selection? Second, once we have rejected the notion of winner-take-all algorithm evaluation, how ought novel algorithms to be evaluated? Taking the idea of boosting from machine learning as our guiding metaphor, we strive to answer both questions.

10.1.1 The Boosting Metaphor

Boosting is a machine learning paradigm due to Schapire [1990] and widely studied since. Although this chapter does not make use of any technical results from the boosting literature, it takes its inspiration from the boosting philosophy. Stated simply, boosting is based on two insights:

1. Poor classifiers can be combined to form an accurate ensemble when the classifiers' areas of effectiveness are sufficiently uncorrelated.

2. New classifiers should be trained on problems on which the current aggregate classifier performs poorly.

In this chapter, we argue that algorithm design should be informed by two analogous ideas:

1. Algorithms with high average running times can be combined to form an algorithm portfolio with low average running time when the algorithms' easy inputs are sufficiently uncorrelated.

2. New algorithm design should focus on problems on which the current algorithm portfolio performs poorly.

Of course the analogy to boosting is imperfect; we discuss differences in Section 10.5.

10.2 Algorithm Portfolios

Our work described in Chapter 9 demonstrated that statistical regression can be used to learn surprisingly accurate algorithm-specific models of the empirical hardness of given distributions of problem instances. Given these techniques, we propose building portfolios of multiple algorithms as follows:

1. Train a model for each algorithm, as described above.
2. Given an instance:
    (a) Compute feature values
    (b) Predict each algorithm's running time using runtime models.
    (c) Run the algorithm predicted to be fastest

Unlike the models discussed in Chapter 9, for constructing algorithm portfolios we use models that predict actual running time rather than log running time. Other transformations of the response variable, including the log transformation, are evaluated in Section 10.3.2. Overall, while we will show experimentally that our portfolios can dramatically outperform the algorithms of which they are composed, our techniques are also deceptively simple. For discussion and comparison with other approaches in the literature, please see Section 10.5.1.

10.2.1 Experimental Results

In this chapter we consider three algorithms for solving the WDP: CPLEX 7.1; GL [Gonen & Lehmann, 2001], a simple branch-and-bound algorithm with CPLEX's LP solver as its heuristic; and CASS. First, we used the methodology described in Chapter

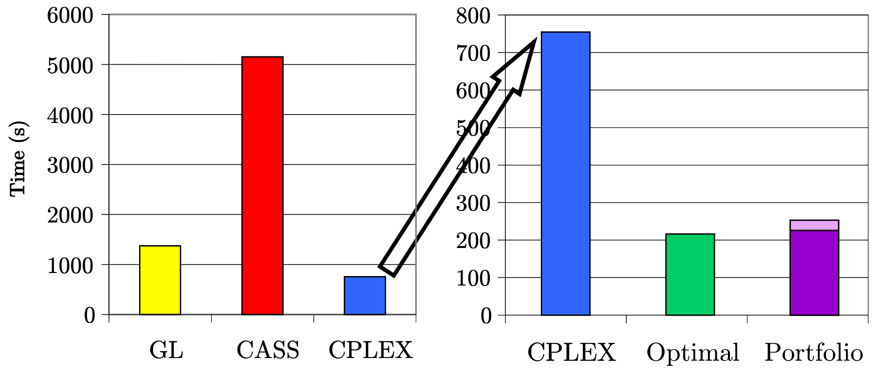


Figure 10.1: Algorithm and Portfolio Runtimes

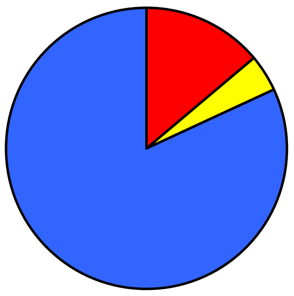


Figure 10.2: Optimal

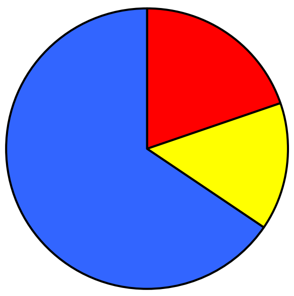


Figure 10.3: Selected

9 to build regression models for GL and CASS. Figure 10.1 compares the average runtimes of our three algorithms (CPLEX, CASS, GL) to that of the portfolio[1]. As we would expect from the results in Section 8.2, CPLEX would be chosen under winner-take-all algorithm selection. The "optimal" bar shows the performance of an ideal portfolio where algorithm selection is performed perfectly and with no overhead. The portfolio bar shows the time taken to compute features (light portion) and the time taken to run the selected algorithm (dark portion). Despite the fact that CASS and GL are much slower than CPLEX on average, the portfolio outperforms CPLEX by roughly a factor of 3. Moreover, neglecting the cost of computing features, our portfolio's selections take only 5% longer to run than the optimal selections.

Figs. 10.2 and 10.3 show the frequency with which each algorithm is selected in the ideal portfolio and in our portfolio. They illustrate the quality of our algorithm

[1]Note the change of scale on the graph, and the repeated CPLEX bar

selection and the relative value of the three algorithms. We know from Figure 8.10 that CASS is often significantly uncorrelated with CPLEX; it turns out that most of the speedup in our portfolio comes from choosing CASS on appropriate instances. Observe that our portfolio does not always make the right choice (in particular, it selects GL much more often than it should). However, most of the mistakes made by our models occur when both algorithms have very similar running times; these mistakes are not very costly, explaining why our portfolio's choices have a running time so close to optimal.

Observe that our variable importance analysis from Chapter 9.4.2 gives us some insight about why an algorithm like CASS is able to provide such large gains over algorithms like CPLEX and GL on a significant fraction of instances. Unlike either CPLEX or GL, CASS does not use an LP heuristic, and so does not incur a substantially higher per-node cost when the number of constraints (and thus the bid graph edge density feature) increases. Like any search algorithm, CASS does benefit whenever the search space becomes smaller; thus, CASS can achieve better overall performance on problems with a very large number of constraints.

These results show that our portfolio methodology can work very well even with a small number of algorithms, and when one algorithm's average performance is considerably better than the others'. We suspect that our techniques could be even more effective in other settings.[2]

10.3 Extending our Portfolio Methodology

Once it has been demonstrated that algorithm portfolios can offer significant speedups over winner-take-all algorithm selection, it is worthwhile to consider modifications to the methodology that make it more useful in practice. Specifically, we describe methods for reducing the amount of time spent computing features, transforming the response variable, and capping runs of some or all algorithms.

[2]Although it is not presented in this thesis, we have had success applying the same techniques to the satisfiability problem, even placing second and third in several categories in the 2003 International SAT Competition.

10.3.1 Smart Feature Computation

Feature values must be computed before the portfolio can choose an algorithm to run. We expect that portfolios will be most useful when they combine several exponential-time algorithms having high runtime variance, and that fast polynomial-time features should be sufficient for most models. Nevertheless, on some instances the computation of individual features may take substantially longer than one or even all algorithms would take to run. In such cases it would be desirable to perform algorithm selection without spending as much time computing features, even at the expense of some accuracy in choosing the fastest algorithm. In order to achieve this, we partition the features into sets ordered by time complexity, $S_1, \ldots, S_l$, with $i > j$ implying that each feature in S_i takes significantly longer to compute than each feature in S_j.[3] The portfolio can start by computing the easiest features, and iteratively compute the next set only if the expected benefit to selection exceeds the cost of computation. More precisely:

1. For each set S_j learn or provide a model $c(S_j)$ that estimates time required to compute it. Often, this could be a simple average time scaled by input size.

2. Divide the training examples into two sets. Using the first set, train models $M_1^i \ldots M_l^i$, with M_k^i predicting algorithm i's runtime using features in $\bigcup_{j=1}^{k} S_j$. Note that M_l^i is the same as the model for algorithm i in our basic portfolio methodology. Let M_k be a portfolio which selects $\arg\min_i M_k^i$.

3. Using the second training set, learn models $D_1 \ldots D_{l-1}$, with D_k predicting the difference in runtime between the algorithms selected by M_k and M_{k+1} based on S_k. The second set must be used to avoid training the difference models on data to which the runtime models were fit.

Given an instance x, the portfolio now works as follows:

4. For $j = 1$ to l

[3]We assume here that features will have low runtime variance. We have found this assumption to hold for the WDP. If feature runtime variance makes it difficult to partition the features into time complexity sets, smart feature computation is more difficult.

(a) Compute features in S_j

(b) If $D_j[x] > c(S_{j+1})[x]$, continue.

(c) Otherwise, return with the algorithm predicted to be fastest according to M_j.

10.3.2 Transforming the Response Variable

Average runtime is an obvious measure of portfolio performance if one's goal is to minimize computation time over a large number of instances. Since our models minimize root mean squared error on predictions of runtime, they appropriately penalize 20 seconds of error equally on instances that take 1 second or 10 hours to run. That is, they penalize the same *absolute error* in the same way regardless of the magnitude of the instance's runtime. However, another motivation is to achieve good *relative error* on every instance regardless of its hardness—we might thus consider that a 20 second error is more significant on a 1 second instance than on a 10 hour instance. Let r_i^p and r_i^* be the portfolio's runtime and the optimal runtime respectively on instance i, let n be the number of instances, and as defined in Equation (3.10) let δ be the Kronecker delta, an indicator function. One measure that gives an insight into the portfolio's relative error is *percent optimal*:

$$\frac{1}{n}\sum_i \delta_{r_i^*}(r_i^p). \tag{10.1}$$

Another measure of relative error is *average percent suboptimal*:

$$\frac{1}{n}\sum_i \frac{r_i^p - r_i^*}{r_i^*}. \tag{10.2}$$

Taking a logarithm of runtime before running the regression allows our minimization of root mean squared error to concentrate more on relative error than on absolute error, roughly equalizing the importance of errors on easy and hard instances. Thus, models that predict a log of runtime achieve better performance in terms of the average percent suboptimal, but worse performance in terms of average runtime. It is also possible to consider other functions between log and linear (identity) that offer different tradeoffs between the relative importance of small and large instances. Figure 10.5 (overleaf) shows linear, log and cube root, one such intermediate function. We found that the cube root function achieved a particularly effective tradeoff on our WDP data: see Section 10.3.4. The functions are normalized by their maximum value, since this does not affect regression, but allows us to better visualize their effects.

10.3.3 Capping Runs

The methodology of Section 10.2 requires gathering runtime data for every algorithm on every problem instance in the training set. While the time cost of this step is fundamentally unavoidable for our approach, gathering perfect data for every instance can take an unreasonably long time. For example, if algorithm a_1 is usually much slower than a_2 but in some cases dramatically outperforms a_2, a perfect model of a_1's runtime on hard instances may not be needed to discriminate between the two algorithms. The process of gathering data can be made much easier by capping the runtime of each algorithm at some maximum and recording these runs as having terminated at the captime. This approach is safe if the captime is chosen so that it is (almost) always significantly greater than the minimum of the algorithms' runtimes; if not, it might still be preferable to sacrifice some predictive accuracy for dramatically reduced model-building time. Note that if any algorithm is capped, it can be dangerous (particularly without a log transformation) to gather data for any other algorithm without capping at the same time, because the portfolio could inappropriately select the algorithm with the smaller captime.

10.3.4 Experimental Results

Fig. 10.4 shows the performance of the smart feature computation discussed in Section 10.3.1, with the upper part of the bar indicating the time spent computing features. Compared to computing all features, we reduce overhead by almost half with nearly no cost in running time.

Table 10.1 shows the effect of our response variable transformations on average

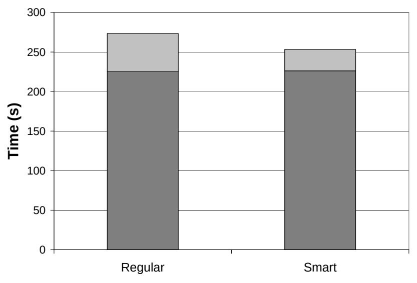


Figure 10.4: Smart Features

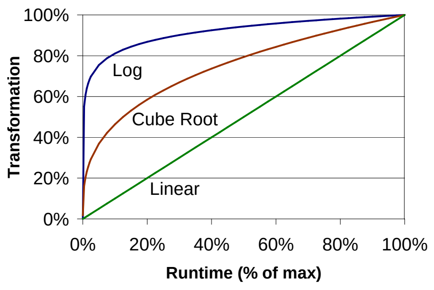


Figure 10.5: Transformation F'ns (normalized)

	Average Runtime	% Optimal	Average % Suboptimal
(Optimal)	216.4 s	100	0
Log	236.5 s	97	9
Cuberoot	225.6 s	89	17
Linear	225.1 s	81	1284

Table 10.1: Portfolio Results

runtime, percent optimal and average percent suboptimal. The first row has results that would be obtained by a perfect portfolio. As discussed in Section 10.3.2, the linear (identity) transformation yields the best average runtime, while the log function leads to better algorithm selection. We tried several transformation functions between linear and log. Here we only show the best, cube root: it has nearly the same average runtime performance as linear, but also made choices nearly as accurately as log.

10.4 Focused Algorithm Design

Once we have decided to select among existing algorithms using a portfolio approach, it makes sense to reexamine the way we design and evaluate algorithms. Since the purpose of designing a new algorithm is to reduce the time that it will take to solve problems, designers should aim to produce new algorithms that complement an existing portfolio. In order to understand what this means it is first essential to choose a distribution D that reflects the problems that will be encountered in practice. Given a portfolio, the greatest opportunity for improvement is on instances that are hard for that portfolio, common in D, or both. More precisely, the importance of a region of problem space is proportional to the amount of time the current portfolio spends working on instances in that region. This is analogous to the principle from boosting that new classifiers should be trained on instances that are hard for the existing ensemble, in the proportion that they occur in the original training set.

10.4.1 Inducing Hard Distributions

Let H_f be a model of portfolio runtime based on instance features, constructed as the minimum of the models that constitute the portfolio. By normalizing, we can reinterpret this model as a density function h_f. By the argument above, we should generate instances from the product of this distribution and our original distribution, D (let $D \cdot h_f(x) = \frac{D(x)h_f(x)}{\int Dh_f}$). However, it is problematic to sample from $D \cdot h_f$: D may be non-analytic (an instance generator), while h_f depends on features and so can only be evaluated after an instance has been created.

One way to sample from $D \cdot h_f$ is *rejection sampling* [Doucet *et al.*, 2001]: generate problems from D and keep them with probability proportional to h_f. This method works best when another distribution is available to guide the sampling process toward hard instances. Test distributions usually have some tunable parameters $\overrightarrow{p}$, and although the hardness of instances generated with the same parameter values can vary widely, $\overrightarrow{p}$ will often be somewhat predictive of hardness. We can generate instances from $D \cdot h_f$ in the following way:[4]

1. Create a new hardness model H_p, trained using only $\overrightarrow{p}$ as features, and normalize it so that it can be used as a pdf, h_p.

2. Generate a large number of instances from $D \cdot h_p$. Observe that we *can* sample from this distribution: h_p is a polynomial, so we can sample from it directly; this gives us parameter values that we can pass to the generator.

[4]In true rejection sampling step 2 would generate a single instance that would be then accepted or rejected in step 3. Our technique approximates this process, but doesn't require us to normalize H_f and allows us to output an instance after generating a constant number of samples.

3. Construct a distribution over instances by assigning each instance s probability proportional to $\frac{H_f(s)}{h_p(s)}$, and select an instance by sampling from this distribution.

Observe that if h_p turns out to be helpful, hard instances from $D \cdot h_f$ will be encountered quickly. Even in the worst case where h_p directs the search away from hard instances, observe that we still sample from the correct distribution because the weights are divided by $h_p(s)$.

In practice, D may be factored as $D_g \cdot D_{p_i}$, where D_g is a distribution over otherwise unrelated instance generators with different parameter spaces, and D_{p_i} is a distribution over the parameters of the chosen instance generator i. In this case it is difficult to learn a single H_p. A good solution is to factor h_p as $h_g \cdot h_{p_i}$, where h_g is a hardness model using only the choice of instance generator as a feature, and h_{p_i} is a hardness model in instance generator i's parameter space. Likewise, instead of using a single feature-space hardness model H_f, we can train a separate model for each generator $H_{f,i}$ and normalize each to a pdf $h_{f,i}$.[5] The goal is now to generate instances from the distribution $D_g \cdot D_{p_i} \cdot h_{f,i}$, which can be done as follows:

1. For every instance generator i, create a hardness model H_{p_i} with features $\overrightarrow{p_i}$, and normalize it to create a pdf, h_{p_i}.

2. Construct a distribution over instance generators h_g, where the probability of each generator i is proportional to the average hardness of instances generated by i.

3. Generate a large number of instances from $(D_g \cdot h_g) \cdot (D_{p_i} \cdot h_{p_i})$

   (a) select a generator i by sampling from $D_g \cdot h_g$

   (b) select parameters for the generator by sampling from $D_{p_i} \cdot h_{p_i}$

   (c) run generator i with the chosen parameters to generate an instance.

[5]However, the experimental results presented in Figures 10.6–10.8 use hardness models H_f trained on the whole dataset rather than using models trained on individual distributions. Learning new models would probably yield even better results.

4. Construct a distribution over instances by assigning each instance s from generator i probability proportional to $\frac{H_{f,i}(s)}{h_g(s) \cdot h_{p_i}(s)}$, and select an instance by sampling from this distribution.

10.4.2 Inducing Realistic Distributions

It is important for our portfolio methodology that we begin with a "realistic" D: that is, a distribution accurately reflecting the sorts of problems expected to occur in practice. Care must always be taken to construct a generator or set of generators producing instances that are representative of problems from the target domain. Sometimes, it is possible to construct a function R_f that scores the realism of a generated instance based on features of that instance; such a function can sometimes encode additional information about the nature of realistic data that cannot easily be expressed in a generator. If a function R_f is provided, we can construct D from a parameterized set of instance generators by using R_f in place of H_f above and learning r_p in the same way we learned h_p. Given these distributions, the techniques described in Section 10.4.1 are guaranteed to generate instances with increased average realism scores.

10.4.3 Experimental Results

Due to the wide spread of runtimes in our composite distribution D (7 orders of magnitude) and the high accuracy of our model h_f, it is quite easy for our technique to generate harder instances. These results are presented in Figure 10.6. Because our runtime data was capped, there is no way to know if the hardest instances in the new distribution are harder than the hardest instances in the original distribution; note, however, that very few easy instances are generated. Instances in the induced distribution came predominantly from the CATS arbitrary distribution, with most of the rest from L3.

To demonstrate that our technique also works in more challenging settings, we sought a different distribution with small runtime variance. As described in Section 7.4.3, there has been ongoing discussion in the WDP literature about whether those

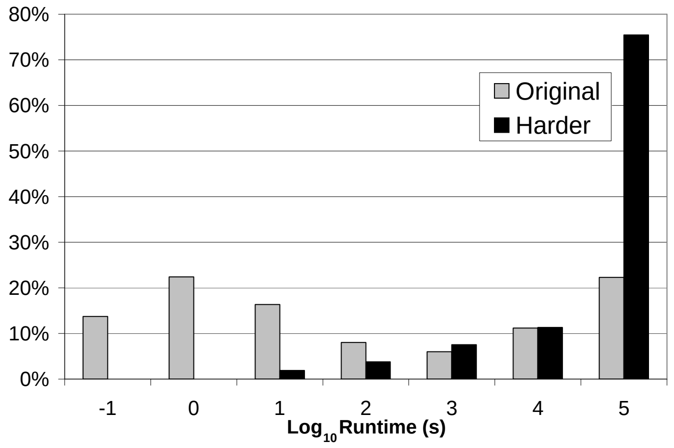


Figure 10.6: Inducing Harder Distributions

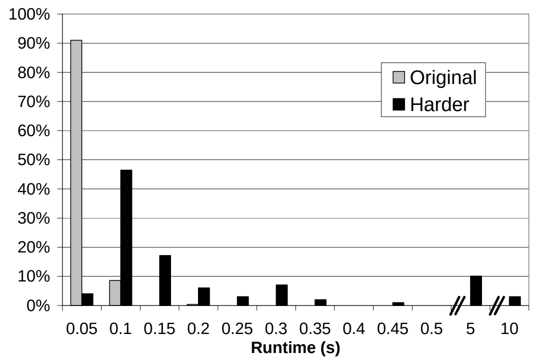


Figure 10.7: Matching

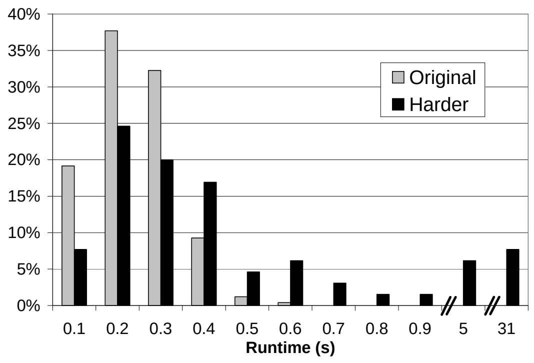


Figure 10.8: Scheduling

CATS distributions that are relatively easy could be configured to be harder. We consider two easy distributions with low variance from CATS, *matching* and *scheduling*, and show that they indeed can be made much harder than originally proposed. Figures 10.7 and 10.8 show the histograms of the runtimes of the ideal portfolio before and after our technique was applied. In fact, for these two distributions we generated instances that were (respectively) 100 and 50 times harder than anything we had previously seen! Moreover, the *average* runtime for the new distributions was greater than the observed *maximum* running time on the original distribution.

10.5 Discussion and Related Work

10.5.1 Algorithm Selection

It has long been understood that algorithm performance can vary substantially across different classes of problems. Rice [1976] was the first to formalize algorithm selection as a computational problem, framing it in terms of function approximation. Broadly, he identified the goal of selecting a mapping $S(x)$ from the space of instances to the space of algorithms, to maximize some performance measure $\text{perf}(S(x), x)$. Rice offered few concrete techniques, but all subsequent work on algorithm selection can be seen as falling into his framework. We explain our choice of methodology by relating it to other approaches for algorithm selection that have been proposed in the literature.

Parallel Execution

One tempting alternative to portfolios that select a single algorithm is the parallel execution of all available algorithms. While it is often true that additional processors are readily available, it is also often the case that these processors can be put to uses besides running different algorithms in parallel, such as parallelizing a single search algorithm or solving multiple problem instances at the same time. Meaningful comparisons of running time between parallel and non-parallel portfolios require that computational resources be fixed, with parallel execution modelled as ideal (no-overhead) task swapping on a single processor. Let $t^*(x)$ be the time it takes to run the algorithm that is fastest on instance x, and let n be the number of algorithms. A portfolio that executes all algorithms in parallel on instance x will always take time $nt^*(x)$. On our WDP data such parallel execution has roughly the same average runtime as winner-take-all algorithm selection (we have three algorithms and CPLEX is three times slower than the optimal portfolio), while our techniques do much better, achieving running times of roughly $1.05t^*(x)$.

In some domains, parallel execution *can* be a very effective technique. Gomes and Selman [2001] proposed such an approach for incomplete SAT algorithms, using the term *portfolio* to describe a set of algorithms run in parallel. In this domain runtime depends heavily on variables such as random seed, making runtime difficult to predict; thus parallel execution is likely to outperform a portfolio that chooses a single algorithm. In such cases it is possible to extend our methodology to allow for parallel execution. We can add one or more new algorithms to our portfolio, with algorithm i standing as a placeholder for the parallel execution of k_i of the original algorithms; in the training data i would be given a running time of k_i times the minimum of its constituents. This approach would allow portfolios to choose to task-swap sets of algorithms in parts of the feature space where the minimums of individual algorithms' runtimes are much smaller than their means, but to choose single algorithms in other parts of the feature space. Our use of the term "portfolio" may thus be seen as an extension of the term coined by Gomes and Selman, referring to a set of algorithms and a strategy for selecting a subset (perhaps one) for parallel execution.

Classification

Since algorithm selection is fundamentally discriminative—it entails choosing the algorithm that will exhibit minimal runtime—classification is an obvious approach to consider. Any standard classification algorithm (*e.g.*, a decision tree) could be used to learn which algorithm to choose given features of the instance and labelled training examples. The problem is that such non-cost-sensitive classification algorithms use the wrong error metric: they penalize misclassifications equally regardless of their cost. We want to minimize a portfolio's average runtime, not its accuracy in choosing the optimal algorithm. Thus we should penalize misclassifications more when the difference between the runtimes of the chosen and fastest algorithms is large than when it is small. This is just what happens when our decision criterion is to select the smallest prediction among a set of regression models that were fit to minimize root mean squared error.

A second classification approach entails dividing running times into two or more bins, predicting the bin that contains the algorithm's runtime, and then choosing the best algorithm. For example, Horvitz *et al.* [2001; 2002] used classification to predict runtime of CSP and SAT solvers with inherently high runtime variance (heavy tails). Despite its similarity to our portfolio methodology, this approach suffers from the use of a classification algorithm to predict runtime. First, the learning algorithm does not use an error function that penalizes large misclassifications (off by more than one bin) more heavily than small misclassifications (off by one bin). Second, this approach is unable to discriminate between algorithms when multiple predictions fall into the same bin. Finally, since runtime is a continuous variable, class boundaries are artificial. Instances with runtimes lying very close to a boundary are likely to be misclassified even by a very accurate model, making accurate models harder to learn.

Markov Decision Processes

Perhaps most related to our methodology is work by Lagoudakis and Littman [2000; 2001]. They worked within the MDP framework, and concentrated on recursive algorithms (*e.g.*, sorting, SAT), sequentially solving the algorithm selection problem on each subproblem. This work demonstrates encouraging results; however, its generality is limited by several factors. First, the use of algorithm selection at each stage of a recursive algorithm can require extensive recoding, and may simply be impossible with 'black-box' commercial or proprietary algorithms, which are often among the most competitive. Second, solving the algorithm selection problem recursively requires that the value functions be very inexpensive to compute; for the WDP we found that more computationally expensive features were required for accurate predictions of runtime. Finally, these techniques can be undermined by non-Markovian algorithms, such as those using clause learning, taboo lists or other forms of dynamic programming. Of course, our approach could also be characterized as an MDP; we do not do so as the framework is redundant in the absence of a sequential decision-making problem.

Different Regression Approaches

Lobjois and Lemaître [1998] select among several simple branch-and-bound algorithms based on a prediction of running time. This work is similar in spirit to our own; however, their prediction is based on a single feature and works only on a particular class of branch-and-bound algorithms.

Since our goal is to discriminate among algorithms, it might seem more appropriate to learn models of pairwise differences between algorithm runtimes, rather than models of absolute runtimes. For linear regression (and the forms of nonlinear regression used in our work) it is easy to show that the two approaches are mathematically equivalent.

10.5.2 Inducing Hard Distributions

It is widely recognized that the choice of test distribution is important for algorithm development. In the absence of general techniques for generating instances that are both realistic and hard, the development of new distributions has usually been performed manually. An excellent example of such work is Selman *et al.* [1996], which describes a method of generating SAT instances near the phase transition threshold, which are known to be hard for most SAT solvers.

10.5.3 The Boosting Metaphor Revisited

Although it is helpful, our analogy to boosting is clearly not perfect. One key difference lies in the way components are aggregated in boosting: classifiers can be combined through majority voting, whereas the whole point of algorithm selection is to run only a single algorithm. We instead advocate the use of learned models of runtime as the basis for algorithm selection, which leads to another important difference. It is not enough for the easy problems of multiple algorithms to be uncorrelated; the models must also be accurate enough to reliably recommend against the slower algorithms on these uncorrelated instances. Finally, while it is impossible to improve on correctly classifying an instance, it is almost always possible to solve a problem instance more quickly. Thus improvement is possible on easy instances as well as on hard instances; the analogy to boosting holds in the sense that focusing on hard regions of the problem space increases the potential gain in terms of reduced average portfolio runtimes.

10.6 Conclusions

Empirical hardness models can be used to combine algorithms together into a portfolio that outperforms its constituents. We have described how to build such portfolios, also presenting techniques for reducing the cost of computing features, reducing the time spent gathering training data by capping runs, and striking the right balance between the penalties for mispredicting easy and hard instances. We argued that algorithm design should focus on problem instances upon which a portfolio of existing algorithms spends most of its time, and provided techniques for using empirical hardness models to automatically induce such distributions.

We performed experiments on WDP algorithms, and showed that a portfolio composed of CPLEX, CASS and GL outperformed CPLEX alone by a factor of 3—despite the fact that CASS and GL are *much* slower than CPLEX on average. We were also able to induce test data that was much harder for our portfolio, and were even able to make specific CATS distributions much harder.

Bibliography

Achlioptas, D., Gomes, C. P., Kautz, H. A., & Selman, B. (2000). Generating satisfiable problem instances. *AAAI.*

Anderson, A., Tenhunen, M., & Ygge, F. (2000). Integer programming for combinatorial auction winner determination. *ICMAS* (pp. 39–46).

Ausubel, L., Cramption, P., McAfee, R., & McMillan, J. (1997). Synergies in wireless telephony: Evidence from the broadband PCS auctions. *Journal of Economics and Management Strategy*, *6*(3), 497–527.

Banks, J., Ledyard, J., & Porter, D. (1989). Allocating uncertain and unresponsive resources: An experimental approach. *RAND Journal of Economics*, *20*, 1–23.

Billings, D., Burch, N., Davidson, A., Holte, R., Schaeffer, J., Schauenberg, T., & Szafron, D. (2003). Approximating game-theoretic optimal strategies for full-scale poker. *IJCAI.*

Blair, D. (1998). Impact of the Internet on core switching network. *Proc. of ENPW'98.* Les Arcs, France.

Blum, B., Shelton, C., & Koller, D. (2003). A continuation method for Nash equilibria in structured games. *IJCAI.*

Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. *Theoretical Aspects of Rationality and Knowledge* (pp. 195–201).

Boutilier, C., Goldszmidt, M., & Sabata, B. (1999). Sequential auctions for the allocation of resources with complementarities. *IJCAI.*

Brewer, P., & Plott, C. (1996). A binary conflict ascending price (BICAP) mechanism for the decentralized allocation of the right to use railroad tracks. *International Journal of Industrial Organization*, *14*, 857–886.

Bykowsky, M., Cull, R., & Ledyard, J. (1995). *Mutually destructive bidding: The FCC auction design problem* (Technical Report Social Science Working Paper 916). California Institute of Technology, Pasadena.

CATS Website (2000). http://robotics.stanford.edu/CATS.

Cheeseman, P., Kanefsky, B., & Taylor, W. M. (1991). Where the Really Hard Problems Are. *IJCAI-91*.

Conitzer, V., & Sandholm, T. (2002). *Complexity Results about Nash Equilibria* (Technical Report CM-CS-02-135). CMU.

Cramton, P., & Palfrey, T. (1990). Cartel enforcement with uncertainty about costs. *International Economic Review*, *31*(1), 17–47.

Crawford, V., & Sobel, J. (1982). Strategic information transmission. *Econometrica*, *50*(6), 1431–1451.

de Vries, S., & Vohra, R. (2003). Combinatorial auctions: A survey. *INFORMS Journal on Computing*, *15*(3).

DeMartini, C., Kwasnica, A., Ledyard, J., & Porter, D. (1998). *A new and improved design for multi-object iterative auctions* (Technical Report Social Science Working Paper 1054). California Institute of Technology, Pasadena.

Demers, A., Keshav, S., & Shenker, S. (1990). Analysis and simulation of a fair queuing algorithm. *Journal of internetworking research and experience*, 3–26.

Doucet, A., de Freitas, N., & (ed.), N. G. (2001). *Sequential monte carlo methods in practice*. Springer-Verlag.

Feinstein, J., Block, M., & Nold, F. (1985). Asymmetric behavior and collusive behavior in auction markets. *American Economic Review*, *75*(3), 441–460.

Floyd, S. (1994). TCP and explicit congestion notification. *ACM Computer Communication Review*, *24*(5), 10–23.

Floyd, S., & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, *1*(4), 397–413.

Friedman, J. (1991). Multivariate adaptive regression splines. *Annals of Statistics*, *19*.

Fudenberg, D., & Tirole, J. (1991). *Game theory*. MIT Press.

Fujishima, Y., Leyton-Brown, K., & Shoham, Y. (1999). Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. *IJCAI*.

Gibbens, R., & Kelly, F. (1999). Resource pricing and the evolution of congestion control. *Automatica*, *35*, 1969–1985.

Gibbens, R., & Key, P. (1999). *The use of games to assess user strategies for differential quality of service in the internet* (Technical Report). Microsoft Research, Cambridge.

Gomes, C., & Selman, B. (2001). Algorithm portfolios. *Artificial Intelligence*, *126*(1-2), 43–62.

Gomes, C. P., & Selman, B. (1997). Problem structure in the presence of perturbations. *AAAI/IAAI*.

Gonen, R., & Lehmann, D. (2000). Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. *ACM Conference on Electronic Commerce*.

Gonen, R., & Lehmann, D. (2001). *Linear programming helps solving large multi-unit combinatorial auctions* (Technical Report TR-2001-8). Leibniz Center for Research in Computer Science.

Graham, D., & Marshall, R. (1987). Collusive bidder behavior at single-object second-price and English auctions. *Journal of Political Economy*, *95*, 579–599.

Graham, D., Marshall, R., & Richard, J.-F. (1990). Differential payments within a bidder coalition and the Shapley value. *American Economic Review*, *80*(3), 493–510.

Grether, D., Isaac, R. M., & Plott, C. (1989). *The allocation of scarce resources: Experimental economics and the problem of allocating airport slots.* Boulder, CO: Westview Press.

Guestrin, C. E., Koller, D., & Parr, R. (2001). Multiagent planning with factored MDPs. *14th Neural Information Processing Systems (NIPS-14)* (pp. 1523–1530). Vancouver, Canada.

Harstad, R., Kagel, J., & Levin, D. (1990). Equilibrium bid functions for auctions with an uncertain number of bidders. *Economic Letters*, *33*(1), 35–40.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *Elements of statistical learning.* Springer.

Hendricks, K., & Porter, R. (1989). Collusion in auctions. *Annales d'Économie et de Statistique*, *15/16*, 216–229.

Holte, R. C. (2001). Combinatorial auctions, knapsack problems, and hill-climbing search. *Canadian Conference on AI.*

Hoos, H., & Boutilier, C. (2000). Solving combinatorial auctions using stochastic local search. *The 17th national conference on artificial intelligence* (pp. 22–29).

Horvitz, E., Ruan, Y., Gomes, C., Kautz, H., Selman, B., & Chickering, M. (2001). A Bayesian approach to tackling hard computational problems. *UAI.*

Hotelling, H. (1929). Stability in competition. *Economic Journal*, *39*, 41–57.

Kastner, R., Hsieh, C., Potkonjak, M., & Sarrafzadeh, M. (2002). On the sensitivity of incremental algorithms for combinatorial auctions. UCLA CS Tech. Report 020000.

Kearns, M., Littman, M., & Singh, S. (2001). Graphical models for game theory. *UAI.*

Kearns, M., & Mansour, Y. (2002). Efficient nash computation in large population games with bounded influence. *UAI.*

Key, P., & McAuley, D. (1999). Differential QoS and pricing in networks: Where flow control meets game theory. *IEE Proc Software 146.*.

Kohavi, R., & John, G. (1997). Wrappers for feature subset selection. *Artificial Intelligence Journal, special issue on relevance*, *97*(1–2), 273–324.

Koller, D., & Milch, B. (2001). Multi-agent influence diagrams for representing and solving games. *IJCAI.*

Korf, R., & Reid, M. (1998). Complexity analysis of admissible heuristic search. *AAAI-98.*

Lagoudakis, M., & Littman, M. (2000). Algorithm selection using reinforcement learning. *ICML.*

Lagoudakis, M., & Littman, M. (2001). Learning to select branching rules in the DPLL procedure for satisfiability. *LICS/SAT.*

Ledyard, J., & Szakaly, K. (1994). Designing organizations for trading pollution rights. *Journal of Economic Behavior and Organization*, *25*, 167–196.

Ledyard, J. O., Porter, D., & Rangel, A. (1997). Experiments testing multiobject allocation mechanisms. *Journal of Economics & Management Strategy*, *6*(3), 639–675.

Lehmann, D., O'Callaghan, L., & Shoham, Y. (1999). Truth revalation in rapid, approximately efficient combinatorial auctions. *ACM Conference on Electronic Commerce.*

Leyton-Brown, K., Nudelman, E., Andrew, G., McFadden, J., & Shoham, Y. (2003a). Boosting as a metaphor for algorithm design. *Constraint Programming.*

Leyton-Brown, K., Nudelman, E., Andrew, G., McFadden, J., & Shoham, Y. (2003b). A portfolio approach to algorithm selection. *IJCAI.*

Leyton-Brown, K., Nudelman, E., & Shoham, Y. (2002a). Learning the empirical hardness of optimization problems: The case of combinatorial auctions. *CP*.

Leyton-Brown, K., Pearson, M., & Shoham, Y. (2000a). Towards a universal test suite for combinatorial auction algorithms. *ACM EC*.

Leyton-Brown, K., Porter, R., Venkataraman, S., & Prabhakar, B. (2001). Smoothing out focused demand for network resources (short paper). *ACM Conference on Electronic Commerce*.

Leyton-Brown, K., Porter, R., Venkataraman, S., Prabhakar, B., & Shoham, Y. (2003c). Incentive mechanisms for smoothing out a focused demand for network resources. *Computer Communications*, *26*, 237–250.

Leyton-Brown, K., Shoham, Y., & Tennenholtz, M. (2000b). An algorithm for multi-unit combinatorial auctions. *Proceedings of AAAI-00*.

Leyton-Brown, K., Shoham, Y., & Tennenholtz, M. (2000c). Bidding clubs: institutionalized collusion in auctions. *ACM Conference on Electronic Commerce*.

Leyton-Brown, K., Shoham, Y., & Tennenholtz, M. (2002b). Bidding clubs in first-price auctions. *The 19th National Conference on Artificial Intelligence*.

Leyton-Brown, K., & Tennenholtz, M. (2003). Local-effect games. *IJCAI*.

Lobjois, L., & Lemaître, M. (1998). Branch and bound algorithm selection by performance prediction. *AAAI*.

MacKie-Mason, J., & Varian, H. (1994). Pricing the internet. In B. Kahin and J. Keller (Eds.), *Public access to the internet*. Prentice-Hall.

Mailath, G., & Zemsky, P. (1991). Collusion in second-price auctions with heterogeneous bidders. *Games and Economic Behavior*, *3*, 467–486.

Mas-Colell, A., Whinston, M. D., & Green, J. R. (1995). *Microeconomic theory*. New York: Oxford University Press.

McAfee, R., & McMillan, J. (1987). Auctions with a stochastic number of bidders. *Journal of Economic Theory*, *43*, 1–19.

McAfee, R., & McMillan, J. (1992). Bidding rings. *American Economic Review*, *82*, 579–599.

Milgrom, P. (1998). Putting auction theory to work: The simultaneous ascending auction. Technical Report 98-0002, Department of Economics, Stanford University.

Mitchell, B. (1978). Pricing policies in selected European telephone systems. *Proceedings of 6th Conference on Telecommunications Policy Research* (pp. 437–475).

Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B., & Troyansky, L. (1998). Determining computational complexity for characteristic 'phase transitions'. *Nature*, *400*.

Monderer, D., & Shapley, L. (1996). Potential games. *Games and Economic Behavior*, *14*, 124–143.

Monderer, D., & Tennenholtz, M. (2000). Optimal Auctions Revisited. *Artificial Intelligence*, *120*(1), 29–42.

Mura, P. L. (2000). Game networks. *UAI*.

Nash, J. (1950). Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, *36*, 48–49.

Nemhauser, G. L., & Wolsey, L. A. (1988). *Integer and combinatorial optimization*. New York, NY: Wiley.

Nisan, N. (2000). Bidding and allocation in combinatorial auctions. *ACM Conference on Electronic Commerce*.

Nisan, N., & Ronen, A. (2000). Computationally feasible VCG mechanisms. *ACM Conference on Electronic Commerce*.

Odlyzko, A. (1997). *A modest proposal for preventing internet congestion* (Technical Report TR 97.35.1). AT&T Research.

Osborne, M., & Pitchik, C. (1987). Equilibrium in Hotelling's model of competition. *Econometrica*, *55*, 911–922.

Osborne, M., & Rubinstein, A. (1994). *A course in game theory.* MIT Press.

Pan, R., Breslau, L., Prabhakar, B., & Shenker, S. (2001). Approximate fairness through differential dropping. *Submitted*.

Pan, R., Prabhakar, B., & Psounis, K. (2000). CHOKe: A stateless active queue management scheme for approximating fair bandwidth allocation. *Proceedings of IEEE INFOCOM 2000* (pp. 942–951).

Parkes, D. C. (1999). iBundle: An efficient ascending price bundle auction. *ACM Conference on Electronic Commerce.*

Plott, C., & Cason, T. (1996). EPA's new emissions trading mechanism: A laboratory evaluation. *Journal of Environmental Economics and Management*, *30*, 133–160.

Quan, D. (1994). Real estate auctions: A survey of theory and practice. *Journal of Real Estate Finance and Economics*, *9*, 23–49.

Rassenti, S., Reynolds, S., & Smith, V. (1994). Cotenancy and competition in an experimental auction market for natural gas pipeline networks. *Economic Theory*, *4*, 41–65.

Rassenti, S., Smith, V., & Bulfin, R. (1982). A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics*, *13*, 402–417.

Reeves, D., & Wellman, M. (2003). Computing equilibrium strategies in infinite games of incomplete information. *Fifth Workshop on Game Theoretic and Decision Theoretic Agents at the 2nd Conference on Autonomous Agents and Multi-Agent Systems.*

Rice, J. R. (1976). The algorithm selection problem. *Advances in Computers*, *15*, 65–118.

Riley, J., & Samuelson, W. (1981). Optimal auctions. *American Economic Review*, *71*, 381–392.

Robinson, M. (1985). Collusion and the choice of auction. *Rand Journal of Economics*, *16*(1), 141–145.

Rosenthal, R. (1973). A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, *2*, 65–67.

Rothkopf, M., Pekeč, A., & Harstad, R. (1998). Computationally manageable combinatorial auctions. *Management Science*, *44*(8), 1131–1147.

Roughgarden, T., & Tardos, E. (2001). *Bounding the inefficiency of equilibria in nonatomic congestion games* (Technical Report TR2002-1866). Cornell, Ithaca.

Ruan, Y., Horvitz, E., & Kautz, H. (2002). Restart policies with dependence among runs: A dynamic programming approach. *CP*.

S. H. Clearwater, e. (1996). *Market-based control: A paradigm for distributed resource allocation*. World Scientific.

Sandholm, T. (1999). An algorithm for optimal winner determination in combinatorial auctions. *IJCAI-99*.

Sandholm, T., & Suri, S. (2000). Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. *AAAI-00*.

Sandholm, T., Suri, S., Gilpin, A., & Levine, D. (2001). CABOB: A fast optimal algorithm for combinatorial auctions. *IJCAI*.

Sanholm, T. (1993). An implementation of the contract net protocol based on marginal cost calculations. *Proceedings of AAAI-93* (pp. 256–262).

Schapire, R. (1990). The strength of weak learnability. *Machine Learning*, *5*, 197–227.

Schuurmans, D., Southey, F., & Holte, R. C. (2001). The exponentiated subgradient algorithm for heuristic boolean programming. *IJCAI-01.*

Selman, B., Mitchell, D. G., & Levesque, H. J. (1996). Generating hard satisfiability problems. *Artificial Intelligence*, *81*(1-2), 17–29.

Shenker, S. (1995). Making greed work in networks: A game-theoretic analysis of switch service disciplines. *IEEE/ACM Transactions on Networking*, *3*, 819–831.

Slaney, J., & Walsh, T. (2001). Backbones in optimization and approximation. *IJCAI-01.*

Songhurst, D., Stamoulis, G., & Stoer, M. (1999). Usage-based charging using effective bandwidths: studies and reality. *Proceedings of the International Teletraffic Congress, ITC-16.*

Tennenholtz, M. (2000). Some tractable combinatorial auctions. Proceedings of AAAI-2000.

Varian, H. R. (1995). Economic mechanism design for computerized agents. *Proceedings of the First Usenix Conference on Electronic Commerce.*

Vickrey, D., & Koller, D. (2002). Multi-agent algorithms for solving graphical games. *AAAI.*

von Ungern-Sternberg, T. (1988). Cartel stability in sealed bid second price auctions. *The Journal of Industrial Economics*, *18*(3), 351–358.

Wellman, M. (1993). A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, *1*, 1–23.

Wellman, M., Greenwald, A., Stone, P., & Wurman, P. (2003). The 2001 trading agent competition. *Electronic Markets*, *13*(1).

Wellman, M., Wurman, P., Walsh, W., & MacKie-Mason, J. (1998). Auction protocols for distributed scheduling. Games and Economic Behavior.

Zhang, W. (1999). *State-space search: Algorithms, complexity, extensions, and applications*. Springer.

Zurel, E., & Nisan, N. (2000). An efficient approximate allocation algorithm for combinatorial auctions. *ACM Conference on Electronic Commerce*.

Index