AFRL-IF-RS-TR-2005-254
Final Technical Report
July 2005

REPRESENTATION AND COMPUTATION WITH DECISION – AND GAME – THEORETIC AGENTS

Stanford University

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. K547

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

AFRL-IF-RS-TR-2005-254 has been reviewed and is approved for publication

APPROVED: /s/
JOHN F. LEMMER
Project Engineer

FOR THE DIRECTOR: /s/
JAMES W. CUSACK, Chief
Information Systems Division
Information Directorate

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 074-0188*

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** July 2005	**3. REPORT TYPE AND DATES COVERED** **Final Jun 00 – Sep 04**

4. TITLE AND SUBTITLE REPRESENTATION AND COMPUTATION WITH DECISION – AND GAME – THEORETIC AGENTS	**5. FUNDING NUMBERS** G - F30602-00-2-0598 PE - 62301E PR - TASK TA - 00 WU - 10
6. AUTHOR(S) Yoav Shoham	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Stanford University Computer Science Department, Artificial Intelligence Labs Gates Building 1A Stanford CA 94305-9010	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency AFRL/IFSA 3701 North Fairfax Drive 525 Brooks Road Arlington VA 22203-1714 Rome NY 13441-4505	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2005-254

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: John F. Lemmer/IFSA/(315) 330-3657 John.Lemmer@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT *APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.*	**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(Maximum 200 Words)***

This project aimed at working towards a theory of representation and computation in decision – and game-theoretic models. In many domains, such as the ones faced by the military, or in e-commerce, task allocation (and other multi-agent protocols) is carried out in non-cooperative environments. Game theory deals with decision making in non-cooperative environments, but ignores fault-tolerance and related computational issues.

We have isolated several central and complementary lines of research, which are essential for establishing a general synthesized theory of representation and computation in decision and game-theoretic models, and made significant progress in each of those directions.

14. SUBJECT TERMS Multi-party computation, multi-agent adaptation, game-theoretic models	**15. NUMBER OF PAGES** 138
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UL

Table of Contents

INTRODUCTION

In this project we have worked towards a theory of representation and computation in decision- and game-theoretic models. In many domains, such as the ones faced by the military, or in e-commerce, task allocation (and other multi-agent protocols) is carried out in non-cooperative environments. Game theory deals with decision making in non-cooperative environments, but ignores fault-tolerance and related computational issues. Conversely, most work done in computer science ignores the issue of decentralized decision making by self-interested parties. In order to effectively and efficiently address the problems such as task allocation, work in computer science and AI should be integrated with work in game theory. Naturally, this suggests major computational challenges that should be addressed as well.

We have isolated several central and complementary lines of research, which are essential for establishing a general synthesized theory of representation and computation in decision and game-theoretic models, and made significant progress in each of those directions.

1. Computational mechanism design:

An important aspect of our research concerns the use of auctions for efficient resource (or, symmetrically, task) allocation. This is an area we and others have worked on for several years now, and is becoming quite popular in computer science. Auctions fall under the umbrella of mechanism design in game theory, which is the art and science of designing interactions among agents that lead - by virtue of the agents following their own best interest - to certain desired outcomes. In an alternative view, mechanism design injects game-theoretic notations necessary for dealing with self-interesting agents into classical computational settings.

1a. Fair mechanism design: Fair Imposition. [1]

Traditional work in auction theory and economic mechanism design does not concentrate on whether the outcome is fair in any sense; the usual yardsticks are efficiency and revenue maximization (or cost minimization). We investigated this new direction, so that for example if the military decides to adopt a market mechanism to assign transportation tasks to civilian carriers then the outcome will be not only efficient but also fair. We defined the problem precisely, present solution criteria for these problems (the central of which is called *k-efficiency*), and present both positive results (in the form of concrete protocols) and negative results (in the form of impossibility theorems) concerning these criteria.

1b. Fault-tolerant mechanism design. [2]

Traditional work on mechanism design assumes that agents have perfect control over their actions and their success. In contrast, a long-standing interest in computer science is in controlling for failures of various kinds - a computer crashing, a communication link going down. We introduced the notion of Fault Tolerant Mechanism Design, which consists of injecting the computer science notion of fault tolerance into the standard game theoretic framework of Mechanism Design. Specifically, we defined the problem of task allocation in the context in which not only the agents' costs are private information, but so are their probabilities of failure. For several different instances of this setting we obtained technical results, including positive ones in the form of specific mechanisms with provable desired properties, and negative ones in the form of impossibility theorems.

1c. Collusion in first-price auctions. [3]

We introduced a class of mechanisms, called *bidding clubs,* which allow agents to coordinate their bidding in auctions. We modeled this setting as a Bayesian game, including agents' choices of whether or not to accept a bidding club's invitation. It turns out, that for this setting in first-price auctions there exists a Bayes-Nash equilibrium where agents choose to participate in bidding clubs when invited and truthfully declare their valuations to the coordinator. Furthermore, the existence of bidding clubs benefits all agents (including both agents inside and outside of a bidding club) in several different senses.

1d. Practical job-scheduling mechanisms. [4]

We considered the problem of online real-time scheduling of jobs on a single processor in an economic setting, in which each job is released to a separate, self-interested agent. The agent can then delay releasing the job to the algorithm, inflate is length, and declare an arbitrary value and deadline for the job, while the center determines not only the schedule, but also the amount each agent must pay. For the resulting mechanism design problem, we developed a mechanism that addresses each of these incentive issues, while only increasing by one the competitive ratio previously known for the non-strategic setting. We also proved a matching lower bound for deterministic mechanisms that never pay the agents.

1e. Non-cooperative computation [5]

We introduced the framework of non-cooperative computation (NCC). We considered functions whose inputs are held by separate, self-interested agents. We also considered four components of each agent's utility function: (a) the wish to know the correct value of the function, (b) the wish to prevent others from knowing it, (c) the wish to prevent others from knowing one's own private input, and (d) the wish to know other agents' private inputs. We provided an exhaustive game-theoretic analysis of all 24 possible lexicographic orderings among these four considerations, for the case of Boolean functions (mercifully, these 24 cases collapse to four). In each case we identified the class of functions for which there exists an incentive-compatible mechanism for computing the function.

This work has applications both to cryptographic settings and to utility elicitation. If a function is not computable in the NCC setting, then no cryptographic protocol can be developed for interactions between self-interested agents. One application to utility elicitation is when the input to the function is each agent's utility for the events it has experienced (e.g., movies seen or books read). The function that each agent is trying to compute could is then a recommendation based on the opinions of all other agents. The key point is that, in order to construct useful protocols for eliciting utility, one has to be aware of agents' meta-preferences expressing their privacy, their desire to deceive, etc.
Both of these applications fall into a new area that we termed *informational mechanism design*, in which the utility functions of the self-interested agents depend only on the state of knowledge of all agents at the end of the mechanism.

1f. Multi-party computation. [6]

We studied the problem of using a "busy center" to design a mechanism that encourages rational actors to play a game of complete information that achieves an outcome that a center prefers without involving the center in the mechanism in any way on the equilibrium path. We examined the cases when agents' actions are both observable and unobservable to the center. We showed that "busy center" mechanisms can be transformed into agent-resolved mechanisms with the assistance of a trusted third-party bank which is ignorant of the mechanism being executed. These ideas are also applicable in a constant-round multi-party rational exchange protocol.

2. Multi-agent adaptation:

Many, indeed most, multiagent interactions do not admit easy equilibrium analysis, and indeed the harder the analysis the less clear it is that the concept of equilibrium is relevant either descriptively or prescriptively. This is particular true of situations that take place over long period times. What is more common in complex situations is for agents to consider a simple version, or aspect, of the game being played, but learn over time to improve their model and strategies. Indeed, effective adaptation is the topic of many research activities in many academic disciplines. Most of the work concerns single-agent learning, in the context of a passive environment; the environment may be stochastic and unpredictable, but it is not "strategic" which its own set of objectives. Things become even more challenging when multiple agents adapt simultaneously, since one agent's adaptation process impacts the other agent's optimal adaptation. Indeed, learning has been a very active area within game theory, and very recently also within computer science (mostly AI). We have tried to extend AI-style reinforcement learning to the multiagent setting.

2a. Planning in games.

We developed a new anytime algorithm for computing good policies in many-agent systems. Most of the previous work on multi-agent systems tackles problems with either two agents or a very large number of agents. We focused on problems with intermediate number of agents. Our first results show how to solve the planning problem when playing against known rationally bounded opponents, where the computation of the optimal policy may be intractable due to the huge state-space. We identified a lattice of problems where at one extreme we consider all the opponents and compute the true best response, and at the other extreme we consider none of them and compute the maxmin strategy. In between the two extremes we can explicitly reason about a given subset of the opponents and treat the others as worst-case adversaries. We identified a vertex on the lattice that gives the best policy given computational restrictions. Based on our experimental results, we also proposed two ways of heuristically speeding up optimal value computations.

2b. Computing best-response strategies via sampling. [7]

We investigated the problem of computing a best-response to an opponent's strategy, when this strategy is not known exactly but can instead be sampled. For instance, a government might capture some members of a terrorist organization and learn samples of their strategies. They can the use this sample to estimate the strategies of the remaining agents. A similar example involves studying the code/strategy of computer viruses and using this information to design effective security against new viruses. We found analytics results on the number of samples required in order to approximate the optimal best response. We then showed experimentally that convergence to the best response often occurs much more quickly than is predicted by formal guarantees. Finally, we went beyond so-called "oblivious sampling", i.e. we considered what happens if the opponent is aware that the agent has taken the samples, if the agent knows that the opponent is aware, and so on to higher levels of mutual modeling.

3. Algorithms and representations for game theory:

Game theory is a very natural tool for modeling multiagent interactions. In our work on mechanism design we have shown how to inject incentives into computational settings in order to make agents behave the way we want. In the work on multiagent adaptation, we have studied how one might devise artificial agents that act in such environments. A complementary direction is to study the problem of automatically, but centrally, reasoning about game-theoretic settings. However, very little work in computer science has been done on algorithms and representations for game-theoretic problems. An important part of our research has been to fill this gap.

3a. Dispersion games. [8]

We generalized a notion of *anti-coordination* games to an arbitrary number of actions and players. In such games players prefer maximally dispersed outcomes. Such games capture many natural problems, such as division of roles within a team, network load balancing and resource allocation, niche selection in economics. We studied empirically and analytically behavior of agents in repeated versions of such games using several decentralized learning rules.

3c. Search methods for finding Nash equilibria. [9]

Computing a sample Nash equilibrium in a normal form game is one of the most interesting and most poorly understood problems in computer science. The question of its complexity still remains open. The best algorithms currently known are based on complicated mathematics and numerical methods, and yet come with no known guarantees on their running time. Surprisingly, none of the classic AI search methods have been tried. We filled this gap, by formulating this problem as a search problem, and by coming up with problem relaxations and useful heuristics. Resulting methods, while extremely simple, outperformed state-of-the-art Lemke-Howson, Simplicial Subdivision, and Govindan-Wilson algorithms by orders of maginuted on a wide variety of realistic game distributions.

3d. Game theory test suite. [10]

In order to achieve our goal of a coherent theory of decision – and game-theoretic models, it is essential to have a solid collection of interesting settings on which the new ideas can be evaluated. In our case, many of such settings can be modeled as simultaneous-action games. We have combed through the literature in game theory, economics, CS, AI, psychology, and political science, and compiled an extensive database of classes of games in normal form that researches have found to be interesting and useful. We have elucidated relations between many such classes by organizing them into a unified taxonomy of games. We also implemented generators for many of these games in a test-suite called GAMUT. GAMUT has already become the definitive test suit for testing game-theoretic algorithms and agents.

3e. Multi-agent target surveillance.

We developed a novel approach to solve a real-time multiagent target surveillance problem. The problem consists of a set of agents and a set of targets on a map, each of which must be visited by an agent. The objective function is the total time spent by the agents. There exists software to solve this problem optimally; however, in general the computational time is large. On the other hand, there are several different algorithms which quickly produce an approximate solution. Our overall objective is to minimize the sum of the computation time plus the execution time. Our approach is to first execute the approximate solutions. Then, using our empirical hardness methodology, we predict both the improvement in solution quality that would result from finding the optimal solution, in addition to the time that it would take to find this solution. Only if it is worth the computation time, do we execute the optimal solver. Experimentally, this approach outperformed both always using the approximate solution, and always finding the optimal solution.

3f. Representation and complexity of cooperative games. [11]

While most of our effort has focused on non-cooperative games, cooperative game theory is an equally important, but much less understood from the computational point of view. Cooperative game theory studies the interaction of agents that have the capability of enforcing contracts, and has strong ties to basic microeconomic theory. We have introduced several natural compact representations of coalitional games, and studied the complexity of reasoning about games in these representations.

4. Empirical hardness models:

Many natural computational problems associated with game theory and economics models turn out to be NP-hard. Nevertheless, it is important to develop methods for solving practical instances of these problems. Thus, it is necessary to be able to empirically understand factors that make particular problems hard for existing algorithms. We have proposed a novel methodology for doing this. In our methodology, we compute features of hard problem instances, and then use machine-learning, specifically, regression, techniques to construct statistical models of algorithms behavior. We have constructed such models for algorithms in two domains: winner-determination problem for Combinatorial Auctions, and the problem of determining satisfiability of a propositional formula. In both cases, we showed that very accurate predictions of runtimes can be obtained. Also, by analyzing learned models, we were able to zero on some very specific, and, sometimes, surprising, sources of hardness. We have also used this methodology as a basis for our multi-agent target surveillance approach discussed below.

4a. Algorithm selection and algorithm portfolios. [12, 13]

A lot of the problems that naturally arise in AI and game theory are hard. It is often the case, especially for NP-hard problems, that different algorithms perform well on completely different problem instances. Therefore, in order to obtain practical robust solution methods, several existing algorithms must be combined into a single portfolio. The problem of selecting the right algorithm on a per-instance basis was first recognized by Rice in 1973. Surprisingly little has been done since, with most people simply running the algorithm that is best on average. We have demonstrated that our empirical hardness models can be used to select the right algorithm very effectively. For the winner determination problem, a portfolio that included two additional inferior algorithms outperformed the best algorithm by a factor of 3. Our SAT portfolio performed very favorably in SAT solver competition, being the only solver that placed near the top in more than one competition track.

4b. Hard instance distributions. [12, 13]

In light of the success of algorithm portfolios, it becomes extremely important to come up with new hard distributions of problem instances, since new algorithms are only useful to the portfolio if they perform well where the current portfolio performs poorly. We have showed how our hardness models, in conjunction with rejection sampling, can be used to generate hard instances. We have tested this approach on CATS – the standard generator for combinatorial auctions WDP instances. Using our models to guide instance generation, even CATS' easiest distributions, such as matching and scheduling, now routinely output instances that are much harder than anything that we've observed previously.

5. List Of Publications

[1] R. Porter, Y. Shoham, and M. Tennenholtz. Fair Imposition. *Journal of Economic Theory*, 2004.

[2] R. Porter, A. Ronen, Y. Shoham, and M. Tennenholz. Mechanism Design With Execution Uncertainty. In *Proc. of the National Conference on Uncertainty in Artificial Intelligence*,
2002.

[3] K. Leyton-Brown, Y. Shoham, M. Tennenholtz. Bidding Clubs in First-Price Auctions. In *Proc. of the National Conference on Artificial Intelligence*, 2002.

[4] R. Porter. Mechanism Design for Online Real-Time Scheduling. In *Proc. of the ACM Conference on Electronic Commerce*, 2004.

[5] R. McGrew, R. Porter, and Y. Shoham. Towards a General Theory of Non-Cooperative Computation. In *Proc. of the Theoretical Aspects of Rationality and Knowledge*, 2003.

[6] R. McGrew, Y. Shoham. Using Contracts To Influence The Outcome of a Game. In *Proc. of the National Conference on Artificial Intelligence*, 2004.

[7] R. Powers and Y. Shoham. Computing Best-Response Strategies via Sampling. Technical Report, 2004.

[8] T. Grenager, R. Powers and Y. Shoham. Dispersion Games: General Definitions and Some Specific Learning Results. In *Proc. of the National Conference on Artificial Intelligence*, 2002.

[9] R. Porter, E. Nudelman and Y. Shoham. Simple Search Methods for Finding a Nash Equilibrium. In *Proc. of the National Conference on Artificial Intelligence*, 2004.

[10] E. Nudelman, J. Wortman, Y. Shoham, K. Leyton-Brown. Run the GAMUT: A Comprehensive Approach to Evaluating Game-Theoretic Algorithms. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

[11] S. Ieong and Y. Shoham. Marginal Contribution Nets: A Compact Representation Scheme for Coalitional Games. In *Proc. of the ACM Conference on Electronic Commerce*, 2004.

[12] K. Leyton-Brown, E. Nudelman, G. Andrew, J. McFadden, and Y. Shoham. Boosting as a Metaphor for Algorithm Design. Short version in *Proc. of the International Conference on Principles and Practice of Constraint Programming*,
2003.

[13] E. Nudelman, A. Devkar, Y. Shoham, K. Leyton-Brown, and H. Hoos. SATzilla: An Algorithm Portfolio for SAT. *Proc. of the International Conference on Theory and Applications of Satisfiability Testing*, 2004.

Fair Imposition

Yoav Shoham and Moshe Tennenholtz [*]
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

We introduce a new notion, related to auctions and mechanism design, called *fair imposition*.[1] In this setting a center wishes to fairly and efficiently allocate tasks among a set of agents, not knowing their cost structure. As in the study of auctions, the main obstacle to overcome is the self-interest of the agents, which will in general cause them to hide their true costs. Unlike the auction setting, however, here the center has the power to impose arbitrary behavior on the agents, and furthermore wishes to distribute the cost as fairly as possible among them. We define the problem precisely, present solution criteria for these problems (the central of which is called *k-efficiency*), and present both positive results (in the form of concrete protocols) and negative results (in the form of impossibility theorems) concerning these criteria.

1 Introduction

Allocation of resources or, isomorphically, of tasks is among the fundamental problems in computer science, operations research, economics, and other scientific and technological disciplines. In a centralized task allocation problem there is a center whose aim is to allocate one or more tasks among several available agents (be they machines, processors, servers, employees, companies, etc.). Several aspects of the situation strongly impact the problem. The two aspects we focus on are:

1. The information available to the center about the agents' costs for performing the tasks.
2. The center's ability to impose tasks and payments on the agents

In classical work on centralized optimization in CS and OR the assumption is that the center has perfect information about and perfect control over the agents; usually payments don't figure in at all, and the center imposes on the agents an optimal protocol computed by the center. In economics and game theory for the most part the opposite obtains, namely the center (auctioneer, procurer) has no knowledge of the agents' private information, and furthermore has no power to enforce any behavior on the agents. Indeed, the individual freedom to decide whether to transact and under what terms to do so lies at the core of what one usually understands a 'market' to mean. For this reason, payments are the primary means of inducing agents to exhibit any sort of behavior in such a setting.

We introduce an intermediate setting in which the center has full power over the agents, but no access to their private information. We furthermore assume that the center is a benign dictator, which wishes not only to achieve the tasks but to do so in a way that is socially fair. The problem is that agents will in general not volunteer correct information that would allow the center to achieve these objectives, and thus the center must resort to incentive engineering of the sort encountered in mechanism design in game theory.

This broad category of problems was motivated by a specific problem encountered by the military, in the Virtual Transportation Company [VTC] project [Godfrey and Mifflin, 2000]. In most if not all democratic countries the state has the power at times of emergency to commandeer aircraft and other resources required to deal with the crisis. Of course, in a democratic country the state recognizes the rights of companies such as airline carriers, and attempts to compensate them for such use. The problem is how to decide which carriers to tap, and how to compensate the parties affected. Ideally, the state would like to achieve the following:

1. Acquire the required types and number of aircraft.
2. Minimize (in some cases, eliminate) its own costs.
3. Minimize the total true costs to the carriers tapped.
4. Distribute the cost fairly among all the carriers, those tapped and not.

[*]The permanent address of the second author is: Faculty of Industrial Engineering and Management, Technion, Haifa 32000, Israel

[1]This work was supported by DARPA grant F30602-00-2-0598-P00001

While our treatment of the problem going forward will be entirely abstract, and indeed make some assumptions that are not consistent with this application (in particular, that the military wishes to pay less than market value for the services rendered), the reader might keep this example in mind throughout the paper.

The rest of the paper is organized as follows. In section 2 we define the procurement problem. The procurement problem is isomorphic to the problem of task allocation (or the auctioning of a good). In section 3 we discuss the fair imposition of tasks. In particular, we introduce the notion of k-efficiency, which is central for the evaluation of procurement protocols in the context of fair allocation. Sections 4–8 present a set of basic results dealing with the feasibility of fair allocation of tasks. We present several upper bounds (by means of impossibility results) on the level of fairness and the minimization of expenses that can be obtained, as well as protocols for fair task allocation. Further discussion of the contribution of our work in the context of general task allocation and protocols for non-cooperative environments can be found in section 9.

2 The procurement problem

In this section we provide the reader with the requisite knowledge of the task allocation/procurement theory. Basic procurement theory has much similarity with basic auction theory. Following the related basic procurement/auction theory literature, we restrict ourselves to a 1-shot, single-item procurements problem. We later (see Section 8) weaken the second restriction, and in future work deal with the first one.

Consider a center who wishes to obtain a particular service, where there are n potential suppliers, or agents, denoted by $1, 2, \ldots, n$ who may supply this service. A procurement protocol is a procedure in which participants submit messages (typically monetary bids) for providing the service. The protocol's rules specify the type of messages, and as a function of the messages submitted by the participants they determine the service provider and the payments to be made by the participants. The payments may be positive, negative, or zero.

Formally, a procurement procedure for n potential participants, $N = \{1, 2, \ldots, n\}$ is characterized by 4 parameters, M, g, c, d, where M is the set of messages, $g = (g_1, \ldots, g_n)$ with $g_i : M^n \to [0, 1]$ for all i and $\sum_{i=1}^n g_i(m) \leq 1$ for all m, and $c = (c_1, \ldots, c_n); d = (d_1, \ldots, d_n)$ with $c_i, d_j : M^n \to R$ for all i, j. The interpretation of these elements is as follows. Participant i submits a message $m_i \in M$. Let $m = (m_1, m_2, \ldots, m_n)$ be a vector of messages, then the organizer conducts a lottery to determine the service provider, in which the probability that i is the winner equals $g_i(m)$. The winner, say j, is paid $c_j(m)$ and every other participant i pays $d_i(m)$. The classical theory of procurement associates a (Bayesian) game with each procurement procedure and analyses the behavior of the agents under the equilibrium assumption, as described below. Each agent has a type, v_i, selected from a set of possible types V. The type of agent i, v_i, is known to it, but might not be known to the other agents. The type v_i should be interpreted as the cost for agent i in providing the required service. A strategy for agent i is a function $b_i : R_+ \to M$, where $b_i(v_i)$ is the message submitted by i when his type is v_i. Each agent i has a utility function u_i. Assuming that the agents submitted the tuple of messages $m = (m_1, m_2, \ldots, m_n)$, the service will be allocated based on g. The utility of agent i will be $c_i(m) - v_i$ if it has been selected as the service provider, and otherwise its utility will be $-d_i(m)$. A tuple of strategies, $b = (b_1, b_2, \ldots, b_n)$ will be called an equilibrium, if for every agent i, b_i is the best response against the strategies of the other agents in b (denoted by b_{-i}). A strategy b_i of agent i will be called a dominant strategy if for any tuple of strategies b_{-i} of the other agents, and for any strategy b'_i of agent i we have that $u_i(b_i, b_{-i}) \geq u_i(b'_i, b_{-i})$.

3 From procurement to fair imposition

In the setup discussed in this paper, the center can force the agents to provide a desired service, as well as monetary payments. However, the center may wish to get the desired service while attempting to minimize the agents' costs.

Definition 1 *Given a procurement setting, with n possible service providers, $N = \{1, 2, \ldots, n\}$, and costs selected from the set V, a procurement protocol (M, g, c, d), where $M = V$ is the set of messages (possibly declared costs), g is the allocation function, c determines the service provider payments, and d determines the other agents' payments, is called* incentive compatible *if for any cost $v_i \in V$ of agent i, its payoff is maximized by sending the message $m_i = v_i$ regardless of the messages of the other agents (i.e., truth revealing is a dominant strategy).*

In the sequel we will restrict ourselves to incentive compatible protocols. Incentive compatible protocols have several desired properties. In particular, they do not build on any rationality assumption on the behavior of other agents. This implies that an agent can refrain from adopting and computing probabilistic information about the other agents' behavior when employing a dominant strategy.

In order to introduce a basic definition of fair imposition of protocols we use the following idea. We wish first to ensure that the center will minimize its expenses due to the performance of the service. Given that, we would like to minimize the expenses of each and every agent. This leads to the following central definition:

Definition 2 *Given a procurement setting S with n agents, a procurement protocol P is called k-*efficient *if the following holds:*

1. *P is incentive compatible.*
2. *For any tuple of costs $(v_1, v_2, \ldots, v_n)$ where v_i is the cost of agent i, we have that:*

(a) *The sum of payments from the center to the agents is non-positive.*

(b) *The cost to agent i is no more than $\frac{v_{[k]}}{n}$, where $v_{[k]}$ is the k order statistics of $\{v_1, v_2, \ldots, v_n\}$.*[2]

Notice that our definition of fairness captures the desire to minimize expenses of each particular agent. Notice that we have required that the center's budget be non-negative. The case where the procurer expects to pay for its services is discussed in Section 7. In the sequel we will denote the agent with the i-th lowest valuation by a_i and its valuation by $v_{[i]}$.

4 1-efficiency

Work on economic mechanism design in game theory [Fudenberg and Tirole, 1991] usually adopts several basic requirements. One of these basic requirements is that protocols should be economically efficient. We will use the following standard definition:

Definition 3 *Given a procurement setting S with n agents, a procurement protocol P will be called economically efficient if it always selects the agent with the lowest cost to serve as the service provider.*

We can now prove:

Lemma 1 *Given a procurement setting S with n agents, a procurement protocol P is 1-efficient only if it is economically efficient.*

Proof (sketch): Assume that the protocol chooses the agent with the second lowest valuation as the service provider. Recall we denote the agent with the i-th lowest valuation by a_i and its valuation by $v_{[i]}$. In order to get 1-efficiency agents $a_1, a_3, a_4, \ldots, a_n$ need to suffer a cost of at most $\frac{1}{n}v_{[1]}$. This implies that if a_2 is the service provider, then it will suffer an expense of at least $v_{[2]} - \frac{n-1}{n}v_{[1]} > \frac{1}{n}v_{[1]}$, which contradicts 1-efficiency. Similar argument holds when the agent who will provide the service is $a_j, j \geq 2$.

The above lemma teaches us that in order to have 1-efficiency we must have economic efficiency. However, we now show:

Lemma 2 *There is no protocol that is both 1-efficient and economically efficient.*

Proof (sketch): In order to have 1-efficiency agents $a_2, a_3, \ldots, a_n$ should suffer an expense of at most $\frac{v_{[1]}}{n}$. Since the center may not have a negative balance, and since we require 1-efficiency, and since the cost of the service for a_1 is $v_{[1]}$ we get that the payments are as follows: each agent $a_i, i \geq 2$ pays exactly $\frac{1}{n}v_{[1]}$ to the center, who collects these payments and transfers that to a_1. However, such incentive compatible efficient protocol, where the sum of payments is 0, does not exist (see [Mas-Colell *et al.*, 1995], page 880).[3]

[2]The k order statistic of a set is the k lowest element in this set.

[3]This is no longer the case if we consider Bayesian equilibrium [d'Aspremont and Gerard-Varet, 1979].

Hence, as we have seen, economic efficiency is essential for 1-efficiency, but 1-efficiency and economic efficiency are contradicting in our setup. Hence, we get:

Theorem 1 *There is no 1-efficient procurement protocol.*

5 2-efficiency

Given the impossibility of 1-efficiency the next desirable alternative is to consider the case of 2-efficiency. As before, we are first interested in the connections between 2-efficiency and economic efficiency. We can show:

Lemma 3 *Given a procurement setting S with n agents, a procurement protocol P is 2-efficient only if it is economically efficient.*

Proof (sketch): Assume that the protocol assigns the good to agent a_2. Notice that in order to obtain 2-efficiency all other agents will have to pay exactly $\frac{1}{n}v_{[2]}$. This implies that agent 2 will suffer an expense of $\frac{1}{n}v_{[2]}$. Hence, a_2 may go ahead and report a cost v', where $v_{[1]} < v' < v_{[2]}$. Consider now the tuple of types $(v_{[1]}, v', v_{[3]}, \ldots, v_{[n]})$. Regardless of who is the agent to be allocated the service for this valuation (whether this is agent a_2 or another agent), 2-efficiency will imply that agent a_2 will suffer an expense of no more than of $\frac{1}{n}v'$. Hence, the above deviation is rational, which contradicts incentive compatibility. It is immediate to see that an allocation of the service to agent $a_j, j \geq 3$ can not lead to 2-efficiency. Hence, the allocation needs to be efficient.

Thus again we'd like to know whether economic efficiency and 2-efficiency are compatible. Unfortunately, at this time we don't, so instead we give a weaker result. Consider the following property defined on procurement protocols.

Definition 4 *Given a procurement setting S with n agents, a procurement protocol is called* unbiased*, if for every tuple of agents' types the payments by all agents who do not provide the service are identical, and the following property* does not *hold:*

- *For any tuple of agents' types, the service provider's expenses are greater than or equal to the expenses of all other agents, and for at least one tuple of types its expense is greater than the expenses of all other agents.*

Thus a biased procurement protocol either favors the agents that do not provide the service, or differentiates among them. We can now show:

Theorem 2 *Given a procurement setting S with n agents, there is no unbiased 2-efficient procurement protocol.*

Proof (sketch): First, given the previous lemma we can restrict our attention to protocols that guarantee efficient allocation. Notice that for any tuple of types, either all agents' expenses are exactly $\frac{1}{n}v_{[1]}$, or there should be at least one agent who suffers expenses of more than $\frac{1}{n}v_{[1]}$ (otherwise the center will suffer losses). Since the protocol is unbiased there must be at least one tuple of types, for which agent a_2 pays $\frac{1}{n}v_{[1]} + \epsilon$, for some $\epsilon > 0$. Assume that for this tuple of types, a_2 reports the cost $v_{[1]} + \delta$, where $\delta > 0$ and $\frac{v_{[1]}+\delta}{n} < \frac{1}{n}v_{[1]} + \epsilon$ (which is satisfied for any $\delta < n\epsilon$). We get that, since the allocation needs to be economically efficient (and therefore this modification will not change the allocation of service) and since the protocol is 2-efficient, such deviation will decrease the expense of agent a_2: consider the behavior of the protocol on the tuple $(v_{[1]}, v_{[1]}+\delta, v_{[3]}, \ldots, v_{[n]})$, we get that agent a_2 will pay no more than $\frac{v_{[1]}+\delta}{n} < \frac{1}{n}v_{[1]}+\epsilon$. This contradicts incentive compatibility. This implies that there is no unbiased 2-efficient protocol.

6 3-efficiency

The negative results of the previous sections teach us that in order to obtain fair imposition of services we need to consider protocols that are at most 3-efficient. In this section we show that this upper bound can be matched by an appropriate protocol. Consider the following protocol.

Fair3:

1. Each supplier is asked to reveal its costs to the center.
2. The task will be allocated to the agent who has announced the lowest cost; this agent will be paid the second lowest cost announced.
3. Each supplier will pay to the center $\frac{1}{n}$ of the second lowest reported cost of the other participants.

Notice that this protocol (as well as protocol Fair3b to be discussed in section 8) makes use of the Groves scheme for mechanism design [Groves, 1973]. We can now show:

Proposition 1 *Given a procurement setting S with n agents, Fair3 is a 3-efficient protocol for that setting.*

Proof (sketch) : Notice that if every agent reports its actual costs, then the payments by a_1 and a_2 are $\frac{v_{[3]}}{n}$, and the payment by $a_j, j \geq 3$ is $\frac{v_{[2]}}{n}$. We need to show that truth revealing is a dominant strategy. However, since the payments here fit the Groves scheme (see [Groves, 1973]) we get that the protocol is also incentive compatible.

As we can see, if we are willing to settle for 3-efficiency, then quite fair task allocation for minimizing the agents' actual expenses can be obtained. The center will be able to obtain the desired behavior without suffering any cost.

7 Almost budget balanced protocols

Given the general results of the previous sections, it may be of interest to see how far we are from obtaining 2-efficiency. In order to address this issue we challenge one of the assumptions of our model, namely that in no case is any cost imposed on the center. Indeed, while in some situations the center should be expected to pay nothing, in many others they expect to bear some costs. In some cases - including, arguably, the VTC domain, which motivated this work –the expectation is that the service providers experience a positive surplus. Here we however explore the intermediate situation in which the center is willing to suffer some expense, but a minimal one.

In order to handle the above issue let us assume that $V = [a, b]$ where $b > a$, i.e. the agents' costs are in between a and b. In many domains, assuming the costs are high, we have that $b - a << a$. For example, the costs to various airlines of providing a given flight might range from \$750K to \$800K. The center be willing to suffer a payment of \$800K-\$750K=\$50K but not of \$750K or more.

Given the above intuition we consider the following definition:

Definition 5 *Given a procurement setting with n agents, a procurement protocol P will be called an almost budget balanced k-efficient protocol, if the following holds:*

1. *P is incentive compatible.*
2. *For any tuple of costs $(v_1, v_2, \ldots, v_n)$ where v_i is the cost of agent i, we have that:*
   (a) *The sum of payments from the center to the agents is at most $v_{[n]} - v_{[1]}$.*
   (b) *The payment by agent i is no more than $\frac{v_{[k]}}{n}$, where $v_{[k]}$ is the k order statistics of $\{v_1, v_2, \ldots, v_n\}$.*

Consider the following protocol:

almost2:

1. Each supplier is asked to reveal its cost to the center.
2. The task will be allocated to the agent who has announced the lowest cost; this agent will be paid the second lowest cost announced.
3. Each supplier will pay to the center $\frac{1}{n}$ of the lowest reported cost of the other participants.

We can now show:

Proposition 2 almost2 *is an almost budget balanced 2-efficient protocol for any given procurement setting.*

Proof (sketch): Let there be n agents in the setting with valuations $v_{[1]}, ..., v_{[n]}$. Since the payment scheme fits the Groves payments (see [Groves, 1973]) we get that the protocol is incentive compatible. In addition, if every agent reports its actual valuation then the payment by a_1 is $\frac{v_{[2]}}{n}$, and the payment by $a_j, j \geq 2$ is $\frac{v_{[1]}}{n}$. The center will pay $\frac{n-1}{n}v_{[2]} - \frac{n-1}{n}v_{[1]} < v_{[n]} - v_{[1]}$.

8 Extension: the imposition of interacting services

Our study in the previous sections has concentrated on the imposition of a single task on a set of agents. If there are several tasks, we can deal with each of those tasks separately, and apply the techniques and results we previously obtained. Although this may be quite appropriate in many cases, one may wish to consider more general extensions. In this section, we briefly consider one extension. We will concentrate on the case of *two interacting service.*

Consider two services, 1 and 2. The cost of performing service j by agent i will be denoted by $v_i(j)$, and the cost of performing both services by agent i will be denoted by $v_i(\{1,2\})$. The fact that there exists some interaction between the services will be captured by the fact that it might be that $v_i(\{1,2\}) \neq v_i(1) + v_i(2)$. For example, a carrier's cost for a pair of flights might be lower than the sum of costs for each of the flights since these flights might refer to two consecutive periods or two consecutive routes (notice the similarity with combinatorial auctions, a popular topic in the recent AI literature [Fujishima *et al.*, 1999; Sandholm, 1999; Tennenholtz, 2000]; in both cases the valuation for a pair of goods might be different from the sum of valuations of these goods).

The procurement setting definition will be now revised to have two services, and n agents with costs/types as above, selected from a set V. For ease of exposition let $V = [0, b]$ for some $b > 0$.

Definition 6 *Given a procurement setting with two services, and with n possible service providers,* $N = \{1, 2, \ldots, n\}$*, where costs for single services and for the pair of services are selected from the set* V*, a procurement protocol* (M, g, c, d) *is a tuple where* $M = V^3$ *is the set of messages, and a message* $m = (m_1, m_2, m_3)$ *declares the costs for service 1,2, and the pair* $\{1,2\}$ *respectively, where g is the allocation function, c determines the payments to the service providers, and d determines the other agents' payments. Such a protocol is called incentive compatible if for every valuation* $v_i(1), v_i(2), v_i(\{1,2\}) \in V$ *of agent i, its payoff is maximized by sending the message* $m_i = (v_i(1), v_i(2), v_i(\{1,2\}))$ *regardless of the messages of the other agents.*

The definition of k-efficiency can be extended in various ways in order to handle the case of two interacting services. We now describe one of these possible extensions.

Definition 7 *Given a procurement setting S with two interacting services, and n agents, a procurement protocol P will be called k-efficient, if the following holds:*

1. *P is incentive compatible.*
2. *For any tuple of costs of the agents we have:*
   (a) *The sum of payments from the center to the agents is non-positive.*
   (b) *The payment by agent i is no more than* $\frac{v_{[k]}(1)+v_{[k]}(2)}{n}$*, where* $v_{[k]}(j)$ *is the k order statistics of* $\{v_1(j), v_2(j), \ldots, v_n(j)\}$

It is easy to extend our infeasibility results for 1-efficiency and for 2-efficiency to the case of two interacting services. As we will now show, the positive result on 3-efficiency can be generalized as well.

Consider the following protocol:

Fair3b:

1. Each supplier is asked to reveal its costs to the center.
2. The tasks will be allocated to the agents who have announced the lowest cost; notice that both tasks can be allocated to the same agent, or the tasks can be allocated to different agents.
3. If a supplier s has been selected to supply both services then he will be paid as follows: an allocation of the lowest cost possible, ignoring this supplier's messages will be calculated, and the supplier s will be paid according to the cost associated with this allocation.
4. If a supplier s has been selected to supply the service x, and another supplier s' has been selected to supply the service y, then s will be paid as in (3), minus the cost associated with supplying y by s'.
5. For each agent i we consider the second lowest declared cost, $c_i(j)$ of the other agents for service j. Agent i will be asked to pay to the center $\frac{c_i(1)+c_i(2)}{n}$.

We can now show:

Proposition 3 *Given a procurement setting S with two interacting services and n agents, Fair3b is a 3-efficient protocol for that setting.*

9 Discussion

Social systems face the challenge of distributing efforts among service providers, in a way that will obtain the society goals, while attempting to minimize costs for the individuals in that society. Therefore, the problem of fair imposition of tasks appears in a variety of domains, and is fundamental to obtaining efficient and fair procurement procedures. In this paper we have introduced a general rigorous setting, where the fair imposition of tasks can be studied. Using this setting, we have provided general results on the fair imposition of services in multi-agent systems.

Our study deals with the problem of task allocation with self-motivated service providers, where the center can enforce agents' actions (e.g. their payments). The complexity of this setting stems from the fact that the participants can try and cheat the center about their private information (e.g. their costs) while the center wishes to keep the allocation *fair* and to minimize the expenses of each individual participant. As a result, our study complements previous work on social laws (e.g. [Moses and Tennenholtz, 1995; Shoham and Tennenholtz, 1995]) and on the imposition of protocols on multi-agent systems (e.g. [Minsky, 1991a; 1991b]), as well as work in information economics (see [Kreps, 1990] for a general discussion). Our work also complements work in Distributed AI dealing with rules on interactions for self-motivated agents (e.g. [Rosenschein and Zlotkin, 1994; Kraus, 1997]), as well as work bridging the gap between Game Theory and Computer Science [Boutilier *et al.*, 1997; Tennenholtz, 1999; Varian, 1995]. Perhaps the most relevant line of research is work on optimal auction design (which is isomorphic to work on optimal design of procurement protocols; see [Wolfstetter, 1996; McAfee and McMillan, 1987; Milgrom, 1987] for overviews). However, our setting, where fairness is the major objective (rather than economic efficiency or center's revenue) and behaviors can be enforced (hence, participation constraints do not apply), distinguishes our line of research from the economic mechanism design literature.

We see the procurement setting as a basic building block for general task allocation in multi-agent systems. Hence, in order to address general task allocation in non-cooperative environments we need to obtain deep understanding of the basic procurement setup. This seems essential for the design of protocols for non-cooperative environments. Needless to say, when facing this fundamental need, basic work in AI dealing with protocols for non-cooperative environments and work on mechanism design in game theory share much in common. The notion of fair imposition and its study are the contributions of this paper to these lines of research.

Much left to be done. In particular, the study of the multi-item (i.e. interacting services) setting should be further developed. In addition, an explicit treatment of repeated procurement situations is a challenge of considerable importance. More specific challenge is to try and come with a biased 2-efficient protocol. We plan to pursue these extensions in our future work. We believe that the study of the fair imposition of tasks is a challenge of fundamental importance to the design of effective multi-agent systems, and hope that others will join us in this effort.

References

[Boutilier *et al.*, 1997] C. Boutilier, Y. Shoham, and M.P. Wellman. Special issue on economic principles of multi-agent systems. *Artificial Intelligence*, 94, 1997.

[d'Aspremont and Gerard-Varet, 1979] C. d'Aspremont and L.A. Gerard-Varet. Incentives and incomplete information. *Journal of Public Economics*, 11(1):25–45, 1979.

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

[Fujishima *et al.*, 1999] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *IJCAI-99*, 1999.

[Godfrey and Mifflin, 2000] G.A. Godfrey and T.L. Mifflin. Virtual transportation company challenge problem. DARPA working paper, http://www.task-program.org, 2000.

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

[Kraus, 1997] S. Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, 94, 1997.

[Kreps, 1990] D. Kreps. *A Course in Microeconomic Theory.* Princeton University Press, 1990.

[Mas-Colell *et al.*, 1995] A. Mas-Colell, M.D. Whinston, and J.R. Green. *Microeconomic Theory.* Oxford University Press, 1995.

[McAfee and McMillan, 1987] R.P. McAfee and J. McMillan. Auctions and bidding. *Journal of Economic Literature*, 25:699–738, 1987.

[Milgrom, 1987] P.R. Milgrom. Auction theory. In T. Bewly, editor, *Advances in Economic Theory: Fifth World Congress.* Cambridge University Press, 1987.

[Minsky, 1991a] N.H. Minsky. The imposition of protocols over open distributed systems. *IEEE Transactions on Software Engineering*, 17(2):183–195, 1991.

[Minsky, 1991b] N.H. Minsky. Law-governed systems. *Software Engineering Journal*, pages 285–302, September 1991.

[Moses and Tennenholtz, 1995] Y. Moses and M. Tennenholtz. Artificial Social Systems. *Computers and Artificial Intelligence*, 14(6):533–562, 1995.

[Rosenschein and Zlotkin, 1994] Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of Encounter.* MIT Press, 1994.

[Sandholm, 1999] T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *IJCAI-99*, 1999.

[Shoham and Tennenholtz, 1995] Y. Shoham and M. Tennenholtz. Social Laws for Artificial Agent Societies: Off-line Design. *Artificial Intelligence*, 73, 1995.

[Tennenholtz, 1999] M. Tennenholtz. Electronic commerce: From game-theoretic and economic models to working protocols. In *IJCAI-99*, 1999.

[Tennenholtz, 2000] M. Tennenholtz. Some tractable combinatorial auctions. Proceedings of AAAI-2000, 2000.

[Varian, 1995] H.R. Varian. Economic mechanism design for computerized agents. Working paper, Berkeley University, 1995.

[Wolfstetter, 1996] E. Wolfstetter. Auctions: An introduction. *Journal of Economic Surveys*, 10(4):367–420, 1996.

Mechanism Design with Execution Uncertainty

Ryan Porter † **Amir Ronen** †‡ **Yoav Shoham** † **Moshe Tennenholtz** †

†Computer Science Department
Stanford University
{rwporter,amirr,shoham,tennenholtz}@robotics.stanford.edu

‡ICSI
UC Berkeley
amirr@icsi.berkeley.edu

Abstract

We introduce the notion of *fault tolerant mechanism design*, which extends the standard game theoretic framework of mechanism design to allow for uncertainty about execution. Specifically, we define the problem of task allocation in which the private information of the agents is not only their costs to attempt the tasks, but also their probabilities of failure. For several different instances of this setting we present technical results, including positive ones in the form of mechanisms that are incentive compatible, individually rational and efficient, and negative ones in the form of impossibility theorems.

1 INTRODUCTION

Recent years have seen much activity at the interface of computer science and game theory, in particular in the area of Mechanism Design, or MD (e.g. (Parkes & Ungar 2000; Boutilier, Shoham, & Wellman 1997; Shoham & Tennenholtz 2001; Nisan & Ronen 2001)). A sub-area of game theory, MD is the science of crafting protocols for self-interested agents, and as such is natural fodder for computer science in general and AI in particular. The uniqueness of the MD perspective is that it concentrates on protocols for non-cooperative agents. Indeed, traditional game theoretic work on MD focuses uniquely on the incentive aspects of the protocols.

A promising application of MD to AI is the problem of task allocation among self-interested agents (see e.g. (Rosenschein & Zlotkin 1994)). When only the execution costs are taken into account, the task allocation problem allows standard mechanism design solutions. However, this setting does not take into consideration the possibility that agents might fail to complete their assigned tasks. When this possibility is added to the framework, existing results cease to apply. The goal of this paper is to investigate robustness to failures in the game theoretic framework in which each agent is rational and self-motivated. Specifically, we consider the design of protocols for agents which have not only private cost functions, but also privately-known probabilities of failure.

What criteria should such protocols meet? Traditional MD has a standard set of criteria for successful outcomes, namely social efficiency (maximizing the sum of the agents' utilities), individual rationality (positive utility for all participants), and incentive compatibility (incentives for agents to reveal their private information). Fault Tolerant Mechanism Design (FTMD) strives to satisfy these same goals; the key difference is that the agents have richer private information (namely probability of failure, in addition to cost). As we will see, this extension presents novel challenges.

It is important to distinguish between different possible types of failure. The focus of this paper is on failures that occur when agents make a full effort to complete their assigned tasks, but may fail. A more nefarious situation would be one in which agents may also fail deliberately when it is rational to do so. While we do not formally consider this possibility, we will revisit it at the end of the paper to explain why our results hold in this case as well. Finally, one can consider the case in which there exist irrational agents whose actions (for example, intentional failures) are counter to their best interests. This is the most difficult type of failure to handle, because the presence of such agents can affect the strategy of rational agents, in addition to directly affecting the outcome. We leave this case to future work.

It is helpful to consider a concrete example. Consider a network of links which are owned by selfish agents

[1]This work was supported in part by DARPA grant F30602-00-2-0598.

(e.g. airline companies), and two distinguished nodes S and T in it. We allow multiple links between nodes so that more than one agent can provide the same service (but only agent can be selected to do so). When an object is routed through a link, the owning agent incurs some cost. In addition, the agent may *fail* (according to some probability) to pass the object across the link (e.g., the object is lost in transit, or not delivered by a strict deadline). The costs and probabilities are *privately* known to their owners. Our goal is to design a mechanism (protocol) that will ensure that objects will be sent from S to T across the network in the most reliable and cost-effective way possible.

To demonstrate the challenges encountered when facing such problems, consider even the simple case in which the network consists of only parallel links between S and T, and costs are all zero. A naïve protocol would ask each agent for their probability, choose the most reliable agent (according to the declarations) and pay her a fixed, positive amount if she succeeds, and zero otherwise. Of course, in this case each agent will report a probability of one in order to selfishly maximize her *own* expected profit.

In this paper we study progressively more complex task-allocation problems. The first problem that we study is one in which there is only one task. We use this setting both to show why standard MD solutions are not applicable and to present our basic technique in the form of a novel mechanism. After extending this technique to handle the case of multiple tasks without dependencies among them, we move to the general case of dependent tasks. Here, we prove an impossibility result when we demand incentive compatibility in dominant strategies, and present a mechanism that solves in equilibrium the case of dependent tasks. Finally, we discuss the use of cost verification to significantly improve the revenue properties of the center.

2 RELATED WORK

The work presented in this paper integrates techniques of economic mechanism design (an introduction to MD can be found in (Mas-Collel, Whinston, & Green 1995, chapter 23)) with studies of fault tolerant problem solving in computer science and AI.

In particular, the technique used in our mechanism is similar to that of the Generalized Vickrey Auction (GVA) (Vickrey 1961; Clarke 1971; Groves 1973) in that it aligns the utility of the agents with the overall welfare. This similarity is almost unavoidable, as this alignment is the only known general principle for solving mechanism design problems. However, because we allow for the possibility of failures, we will need to change the GVA in a significant way in order for our mechanism to achieve this alignment.

Because we have added probabilities to our setting, our mechanisms may seem to be related to the Expected Externality Mechanism (or d'AGVA) (d'Aspremont & Gerard-Varet 1979), but there are key differences. In the setting of d'AGVA, the types of the agents are drawn (independently) from a distribution which is assumed to be common knowledge among the participants. The two key differences in our setting are that no such common knowledge assumption is made and that the solution concepts which we guarantee are stronger than that of d'AGVA.

A recent paper (Eliaz 2002) also considers failures in MD, but solves a different problem. This work assumes that agents know the types of all other rational agents and also limits the failures that can occur by bounding the number of irrational agents.

Finally, the design of protocols which are robust to failures has a long tradition in computer science (for a survey, see (Linial 1994)). Work in this area, however, almost always assumes a set of agents that are by and large cooperative and adhere to a central protocol, except for some subset of malicious agents who may do anything to disrupt the protocol. In MD settings, the participants fit neither of these classes, but are simply self-interested.

3 A BASIC MODEL

In this section we describe our basic model and notation, which will be modified later to handle specific settings.

In a FTMD problem, we have a set of t tasks $\tau = \{1, \ldots, t\}$ and a set $N = \{1, \ldots, n\}$ of self-interested agents to which the tasks can be assigned. We also have a center M who assigns tasks to agents and pays them for their work. The center and the agents will collectively be called the participants.

Each agent i has, for each task j, a *probability* $p_{ij} \in [0, 1]$ of successfully completing task j, and a nonnegative cost $c_{ij} \in \Re^+$ of attempting the task. We assume that the cost of attempting a task does not depend on the success of the attempt. We use $p_i = (p_{i1}, \ldots, p_{it})$ for the set of all probabilities for agent i, and use $p = (p_1, \ldots, p_n)$ to represent the set of probability vectors for all agents. We use corresponding notation for c_i and c. The pair $\theta_i = (p_i, c_i)$ is called the agent's *type* and is *privately* known to the agent. Each agent is assigned a set A_i of tasks, and her cost to attempt the set is: $c_i(A_i) = \sum_{j \in A_i} c_{ij}$. We define $\theta = (\theta_1, \ldots, \theta_n)$ as the vector of types for all agents.

We use a completion vector $\mu \in \{0, 1\}^t$ to denote which tasks have been completed. The function $V : \{0,1\}^t \rightarrow \Re^+$ defines the center's nonnegative valuation for each possible completion vector. For now, we assume that the center has a non-combinatorial valuation for a set of tasks. That is, the value of a set of tasks is the sum of the values for the individual tasks. We also assume that $V(\mu) \geq 0$ for all μ and that $V(0,\ldots,0) = 0$.

An assignment vector $A = (A_1,\ldots,A_n)$ and a vector of agent probabilities p together induce a probability distribution over the completion vector which we denote by $[\mu|A,p]$. Given an assignment A, a type vector θ and a completion vector μ, we define the *welfare* of the participants as $W(A,c,\mu) = V(\mu) - \sum_i c_i(A_i)$. We define the *expected welfare* as $\bar{W}(A,c,p) = E_{[\mu|A,p]}[W(A,c,\mu)]$. The goal of the center is to design a mechanism (protocol) that maximizes this *expected welfare.*

We assume that each task can be assigned only once. The center does not have to allocate all the tasks. For notational convenience we assume that all the non-allocated tasks are assigned to a dummy agent 0 which for each task has zero probability of success and zero cost to attempt.

When an agent i is assigned a set A_i of tasks, and is paid R_i, her *utility* equals $u_i = R_i - c_i(A_i)$. Since our setting is stochastic by nature, an agent can do no better than to maximize her *expected utility,* $\bar{u}_i$, calculated before any task is attempted. This term thus depends on the true probabilities of success of the agents, as explained below.

Throughout the paper we shall use the following vector notations: The subscript $-i$ on a vector denotes that the term for agent i has been omitted from the vector. For example, $p_{-i} = (p_1,\ldots,p_{i-1},p_{i+1},\ldots,p_n)$. The omitted term can be combined with such a vector by using the following notation: $p = (p_i,p_{-i})$. We denote by μ_i the completion vector for agent i (i.e. we have 1 for each task accomplished by agent i and 0 for each one either failed by her or not assigned to her). The definitions for μ_{-i} and (μ_i,μ_{-i}) follow similarly. Sometimes we will use μ_i in place of p_i. Since both vectors are of the same form, a 0 or 1 for task t_j in μ_i becomes the probability of successfully completing t_j.

3.1 Mechanisms

A mechanism is a protocol that decides how to assign the tasks to the agents and how much each agent is paid. The simplest type of mechanisms are ones in which the agents are simply required to report their types. (Of course they may lie!) The revelation principle (see e.g. (Mas-Collel, Whinston, & Green 1995, p. 871)) tells us that we can, w.l.o.g., restrict ourselves to such mechanisms.

We denote by the vector $\hat{\theta}$ the types declared by the agents . A mechanism is thus defined by a pair $g = (A(\hat{\theta}), R(\hat{\theta},\mu))$ such that:

- $A(\hat{\theta}) = (A_1(\hat{\theta}),\ldots,A_n(\hat{\theta}))$ is an assignment function. It takes a declaration vector and returns an assignment of the tasks to the agents.
- $R(\hat{\theta}) = (R_1(\hat{\theta},\mu),\ldots,R_n(\hat{\theta},\mu))$ is the payment function.

In our motivating example, a type θ_i would correspond to agent i's costs and probabilities of success on each of her edges.

In our protocol, the center first asks each agent to declare her type. We call an agent *truthful* if she reveals her true type to the center. Based on these declarations the center first computes the assignment $A(\hat{\theta})$. Then, the agents execute their tasks. Finally, the center pays the agents. Note that these payments depend on the set of tasks which were accomplished. We assume that the agents always attempt each task to which they are assigned. In our discussion section, we explain why this is a valid assumption.

In the above protocol, the utility of agent i is: $u_i(c_i,\hat{\theta}_i,\hat{\theta}_{-i},\mu) = R_i(\hat{\theta},\mu) - c_i(A_i(\hat{\theta}))$, and her expected utility is: $\bar{u}_i(c_i,\hat{\theta}_i,\hat{\theta}_{-i},p) = E_{[\mu|A(\hat{\theta}),p]}[u_i(c_i,\hat{\theta}_i,\hat{\theta}_{-i},\mu)]$.

The main difference between mechanism design problems and the usual algorithmic problems is that the participating agents may *manipulate* the given protocol if it is *beneficial* for them to do so. We therefore need to design protocols that fulfill our objectives even though the agents behave selfishly. We thus require our mechanism to satisfy the following standard properties:

Individual rationality (IR) holds when truthful agents are guaranteed to have non-negative expected utility. Formally, for all i, θ and $\hat{\theta}_{-i}$: $\bar{u}_i(c_i,\theta_i,\hat{\theta}_{-i},p) \geq 0$.

Incentive compatibility (IC) holds when it is a dominant strategy for each agent to declare her type truthfully. Formally, this condition holds, when for all i, θ, θ'_i, and $\hat{\theta}_{-i}$: $\bar{u}_i(c_i,\theta_i,\hat{\theta}_{-i},p) \geq \bar{u}_i(c_i,\theta'_i,\hat{\theta}_{-i},p)$. This means that the expected utility of the agent (conditional on her own probability of success) is maximized when the agent reports her true type.

A mechanism is called *socially efficient (SE)* if the chosen assignment maximizes the expected welfare $\bar{W}$. The fact that $\bar{W}$ depends on the true types of the agents underscores the importance of IC, which allows the center to correctly assume that $\hat{\theta} = \theta$.

Individual rationality for the center (CR) holds if the *center's utility* $u_M = V(\mu) - \sum_i R_i(.)$ is *always* nonnegative. CR is an extension of the standard mechanism design requirement of weak budget balance to account for the center's utility for outcomes.

A final goal is *no free riders (NFR)*, which holds if all agents not assigned any task have a revenue of zero.

4 SINGLE TASK SETTING

We will start with the special case of a single task to show our basic technique to handle the possibility of failures in MD. For expositional purposes, we will analyze two restricted settings (the first restricts probabilities of success to be one, and the second restricts costs to be zero), before formally proving properties about our mechanism in the full single task setting.

Because there is only one task, we can simplify the notation. We let c_i and p_i denote c_{i1} and p_{i1}, respectively. Similarly, we let $V = V((1))$, which is the value that the center assigns to the completion of the task. For each mechanism, we will use the index [1] to denote the agent selected to attempt the task (e.g., $p_{[1]}$ denotes the selected agent's probability of success). The subscript [2] then refers to the agent who would be selected as the service provider if agent [1] had not participated.

4.1 CASE 1: ONLY COSTS

When we do not allow for failures (that is, $\forall i\ p_i = 1$), the goal of social efficiency reduces to assigning the task the lowest-cost agent. This simplified problem can be solved using a second-price auction (which is a specific case of GVA). Each agent declares a cost, the task is assigned to the agent with the lowest cost, and that agent is paid the second-lowest submitted cost.

4.2 CASE 2: ONLY FAILURES

We now reduce the problem in a different way, by assuming all costs to be zero ($\forall i\ c_i = 0$). In this case, the main goal is to allocate the task to the most reliable agent. Interestingly, we cannot use a straightforward application of the GVA for this case. Such a mechanism would ask each agent to declare a probability of success and assign the task to the agent with the highest declared probability. It would set the revenue function for all agents not assigned the task to be 0, while the service provider would be paid the amount by which her presence increases the welfare of the other agents and the center: $\hat{p}_{[1]}V - \hat{p}_{[2]}V$. Obviously, such a mechanism is not incentive compatible, because the payment to the service provider depends on her own declared type! Since there are no costs, each agent would have a dominant strategy to declare her probability of success as one. [2]

Thus, we need to fundamentally alter our payment rule so that it depends on the outcome of the attempt, and not solely on the declared types, as it does in the GVA. The key difference in our setting that forces this change is the fact that the true type of an agent now directly affects the outcome, whereas in the standard MD setting the type of an agent only affects her preferences over outcomes. We accomplish our goals by replacing $\hat{p}_{[1]}$ with an indicator function that is 1 if the task was completed, and 0 otherwise. The payment rule for the service provider is now $V - \hat{p}_{[2]}V$ if she succeeds and $-\hat{p}_{[2]}V$ if she fails. Just as in the previous setting, the service provider is the only agent who has positive utility for attempting the task with the corresponding payment rule. The expected utility for agent i would be $V \cdot (p_i \cdot (1 - \hat{p}_{[2]}) - (1 - p_i) \cdot \hat{p}_{[2]})$. This expression is positive for the agent iff $p_i > \hat{p}_{[2]}$, which is only true for the service provider.

4.3 CASE 3: COSTS AND FAILURES

We now consider the case of one task with both costs and failures.

We introduce the following definition that we will use throughout the paper: Given an agent i we denote by $\bar{W}^*_{-i}(\hat{c}_{-i}, \hat{p}_{-i})$ the optimal (declared) expected welfare when tasks cannot be allocated to agent i. In the single task case it is $max_{j \neq i}(\hat{p}_j \cdot V - \hat{c}_j)$. Now we can define the mechanism.

Single Task Mechanism:

Assignment The mechanism chooses an agent $i \in \{0, \dots, n\}$ that maximizes the (declared) expected welfare $\bar{W} = \hat{p}_i \cdot V - \hat{c}_i$.

Payment The payment to all agents not assigned a task is always zero. The payment to the "winner" i is defined as follows:

$$R_i = \begin{cases} V - \bar{W}^*_{-i}(\hat{c}_{-i}, \hat{p}_{-i}) & \text{If } i \text{ succeeds} \\ -\bar{W}^*_{-i}(\hat{c}_{-i}, \hat{p}_{-i}) & \text{If } i \text{ fails} \end{cases}$$

Using $p_{[2]}$ and $c_{[2]}$ to denote the probability and the cost of the "second best" agent, the payment to agent i when she succeeds is $(V - p_{[2]} \cdot V + c_{[2]})$ and when she fails is $(-p_{[2]} \cdot V + c_{[2]})$. Note that $\bar{W}^*$ is never negative

[2]In fact, this would be true for any payment rule for which an agent's payment is always nonnegative, which is the reason why we require our goals (such as IC and IR) to be satisfied for the expected utility of the agent, and not for ex post utility.

because the center will never make an assignment that yields an expected loss for the system.

For example, suppose we have three agents with the types listed in Table 1. Let V be 210. If the agents are truthful, then the winner is agent 3. If agent 3 did not exist, the optimal expected welfare would be $\bar{W}^*_{-3} = 210 - 100 = 110$, because the task would be assigned to agent 2. The payment for agent 3 is therefore $210 - 110 = 100$ if she succeeds and -110 if she fails. Agent 3's own costs are 60, and thus her expected utility is $(100 - 60) \cdot 0.9 + (-110 - 60) \cdot 0.1 = 19$.

Agent	c_i	p_i
1	30	0.5
2	100	1.0
3	60	0.9

Table 1: A Single Task Example

Before we prove the properties of this mechanism, let us introduce two definitions that we shall use throughout that paper. Given an agent i, we define the welfare of the other participants $W_{-i}(A, c_{-i}, \mu) = V(\mu) - \sum_{j \neq i} c_j(A_j)$. Note that $W_{-i}(A, c_{-i}, \mu) = W(A, c_i, \mu) + c_i(A_i)$. We define the expected welfare for the other participants as $\bar{W}_{-i}(A, c_{-i}, (p_{-i}, \mu_i)) = E_{[\mu_{-i}|A, p_{-i}]}[W_{-i}(A, c_{-i}, (\mu_i, \mu_{-i}))]$. This is the expected welfare of all the other participants (including the center) when the allocation is A and agent i has completed exactly the set of tasks defined by μ_i.

It is not difficult to see that the payment R_i of each agent i equals $\bar{W}_{-i}(A, \hat{c}_{-i}, (\hat{p}_{-i}, \mu_i)) - \bar{W}^*_{-i}(\hat{c}_{-i}, \hat{p}_{-i})$. Her expected utility is therefore $\bar{u}_i = -c_i(A_i) + E_{[\mu_i|A_i, p_i]}\bar{W}_{-i}(A, \hat{c}_{-i}, (\hat{p}_{-i}, \mu_i)) - \bar{W}^*_{-i}(\hat{c}_{-i}, \hat{p}_{-i})$. Since the distribution $[\mu|A, (p_i, \hat{p}_{-i})]$ equals $[\mu_i|A_i, p_i] \cdot [\mu_{-i}|A, p_{-i}]$, we get that $R_i = \bar{W}(A, (c_i, \hat{c}_{-i}), (p_i, \hat{p}_{-i})) - \bar{W}^*_{-i}(\hat{c}_{-i}, \hat{p}_{-i})$. This means that each agent gets paid her *contribution* to the expected welfare of the other participants.

Theorem 1 *The Single Task mechanism satisfies IR, IC, CR, SE, and NFR.*

Proof:

We will prove each property separately.

1. **Individual Rationality**

   We need to prove that the expected utility of a truthful agent is always non negative. When agent i is truthful her expected utility is $\bar{u}_i = \bar{W}(A((\theta_i, \hat{\theta_{-i}})), (c_i, \hat{c}_{-i}), (p_i, \hat{p}_{-i})) - \bar{W}^*_{-i}(\hat{c}_{-i}, \hat{p}_{-i})$. By the optimality of $A(.)$, the second term quantifies the optimal welfare that can be obtained when the types of the other agents are $\hat{\theta_{-i}}$ and i does not exist. Similarly, the first term quantifies the optimal welfare when the types of the other agents are $\hat{\theta_{-i}}$ and the type of agent i is the true one θ_i. Since i can only improve the total welfare, we proved our claim.

2. **Incentive Compatibility**

   We need to prove that the expected utility of each agent i is maximized when she is truthful. Let $\hat{\theta_{-i}}$ denote the declarations of the other agents. As before, when the agent is truthful her utility is $\bar{u}_i = \bar{W}(A((\theta_i, \hat{\theta_{-i}})), (c_i, \hat{c}_{-i}), (p_i, \hat{p}_{-i})) - \bar{W}^*_{-i}(\hat{c}_{-i}, \hat{p}_{-i})$. Consider the case where the agent reports another type θ'_i. This results in an assignment A'. The utility of agent i in this case is $\bar{u}'_i = \bar{W}(A', (c_i, \hat{c}_{-i}), (p_i, \hat{p}_{-i})) - \bar{W}^*_{-i}(\hat{c}_{-i}, \hat{p}_{-i})$

   Assume by contradiction that $\bar{u}'_i > \bar{u}_i$. This means that $\bar{W}(A', (c_i, \hat{c}_{-i}), (p_i, \hat{p}_{-i})) > \bar{W}(A((\theta_i, \hat{\theta_{-i}})), (c_i, \hat{c}_{-i}), (p_i, \hat{p}_{-i}))$. However this contradicts the optimality of $A((\theta_i, \hat{\theta_{-i}}))$.

3. **Individual Rationality for the Center** Let agent i be the winner. We need to show that the utility for the center is always non negative. There are two cases. When agent i succeeds we have $u_M = V - (V - \bar{W}^*_{-i}(\hat{c}_{-i}, \hat{p}_{-i})) = \bar{W}^*_{-i}(\hat{c}_{-i}, \hat{p}_{-i})) \geq 0$. When i fails, the value is zero and thus $u_M = \bar{W}^*_{-i}(\hat{c}_{-i}, \hat{p}_{-i})) \geq 0$.

4. **Social Efficiency** Immediate from IC and the definition of $A(.)$.

5. **No Free Riders** Immediate from the definition of the payment rule. ∎

5 MULTIPLE TASKS

We now return to the original setting presented in this paper, consisting of t tasks for which the center has a non-combinatorial valuation (that is, the value for a set of tasks is equal to the sum of the values for the individual tasks). Because the setting disallows any interaction between tasks, we can construct a mechanism that satisfies all of our goals by generalizing the Single Task Mechanism.

Multiple Task Mechanism:

Assignment The chosen assignment A maximizes the (declared) expected welfare $\bar{W}(A, \hat{c}, \hat{p}) = E_{[\mu|A, \hat{p}]}[W(A, \hat{c}, \mu)]$.

Payment The payment to each agent i is defined according to her completion vector: $R_i = \bar{W}_{-i}(A, \hat{c}_{-i}, (\hat{p}_{-i}, \mu_i)) - \bar{W}^*_{-i}(\hat{c}_{-i}, \hat{p}_{-i})$

In other words, each agent is paid according to her contribution to the welfare of the other participants.

Proposition 2 *The Multiple Task mechanism satisfies IC, IR, SE, CR, and NFR.*

The proof is similar to the single task case and is omitted. Note that the theorem holds even when the cost functions and probabilities of success have a combinatorial nature.

5.1 COMBINATORIAL V

We now consider the setting in which the center's valuation $V(\cdot)$ can be any monotone function of the tasks. Unfortunately, in this setting, it is impossible to satisfy all our goals simultaneously.

Theorem 3 *When V is combinatorial, there does not exist a mechanism that satisfies IC, IR, CR, and SE for any $n \geq 2$ and $t \geq 2$.*

Intuitively it is enough to consider the following case which no mechanism is able to solve. There are two tasks, each of which can only be completed by one agent (and, this one agent is different for the two tasks). The center only has a positive value (call it x) for both tasks being completed. Since both agents add a value of x to the system, they can each extract close to x from the center, causing the center to pay double for the utility of x he will gain from the completion of the task. Due to space constraints, we omit the formal proof of this theorem.

However, despite the possibility of failures we can maintain the desired properties other than CR using the same mechanism as before.

Theorem 4 *The Multiple Task mechanism satisfies IC, IR, SE, and NFR, even when V is combinatorial.*

Again, we omit the proof. Intuitively, IC, IR, and NFR are not affected by a combinatorial V because they are only properties of the agents, and SE still follows from IC and the definition of $A(\cdot)$.

5.2 DEPENDENCIES

We now return to the case of non-combinatorial valuation $V(\cdot)$, and analyze a different extension: dependencies between the tasks.

Consider our motivating example of a network of flights. A natural example of a task dependency would be an object that could not be carried over the edge (b, c) before being carried over (a, b).

Formally, we say that a task j is *dependent* on a set s of tasks if j cannot be attempted unless all tasks in s were successfully finished. We assume that there are no dependency cycles. The tasks now are executed according to a topological order. Note that if a task cannot be attempted, the agent assigned that task does not incur the costs of attempting it. [3]

However, the presence of dependencies, just like the presence of a combinatorial V, makes it impossible to satisfy IC, IR, CR, and SE.

Theorem 5 *When dependencies exist between tasks, there does not exist a mechanism that satisfies IC, IR, CR, and SE for any $n \geq 2$ and $t \geq 2$.*

Proof: Proof by induction. We first show that a mechanism cannot satisfy IC, IR, CR, and SE for the base case of $n = t = 2$. The inductive step then shows that increasing either n or t cannot alter this impossibility result.

Base Case: We prove the base case by contradiction. Assume that there exists a mechanism that satisfies IC, IR, CR, and SE. This implies that it satisfies these properties for all possible instances, where we define a instance as a particular set of agent types and declarations, task dependencies, and a V function. We will use 3 possible instances in order to derive properties that must hold in the mechanism, but lead to a contradiction. The constants in these instances are that task 2 is dependent on task 1 and that the center has value of 5 for task 2 being completed, but no value for the completion of task 1 in isolation. The five types that we will use, θ_1, θ_1', θ_1'', θ_2, and θ_2', are defined in Table 2 (the final type, θ_e, will be used later in the inductive step).

θ_1 :	$p_{11} = 1$	$c_{11} = 2$	$p_{12} = 1$	$c_{12} = 1$
θ_1' :	$p_{11}' = 1$	$c_{11}' = 2$	$p_{12}' = 0$	$c_{12}' = 0$
θ_1'' :	$p_{11}'' = 1$	$c_{11}'' = 0$	$p_{12}'' = 1$	$c_{12}'' = 4$
θ_2 :	$p_{21} = 0$	$c_{21} = 1$	$p_{22} = 0$	$c_{12} = 0$
θ_2' :	$p_{21}' = 1$	$c_{21}' = 1$	$p_{22}' = 0$	$c_{22}' = 0$
θ_e :	$p_{e1} = 0$	$c_{e1} = 1$	$p_{e2} = 0$	$c_{e2} = 1$

Table 2: Agent Types for Theorem 5.

Instance 1: The true types are θ_1 and θ_2, and the declared types are θ_1 and θ_2'. To satisfy SE, task 1 is assigned to agent 2, and task 2 to agent 1. That is, $A_1(\theta_1, \theta_2') = (0, 1)$ and $A_2(\theta_1, \theta_2') = (1, 0)$. Since agent 2's true type is θ_2, she will fail on task 1, preventing task 2 from being attempted. Thus, $\mu = (0, 0)$ with probability 1. The expected utility for agent 1 is then:

$$\bar{u}_1(c_1, \theta_1, \theta_2', p) = R_1((\theta_1, \theta_2'), (0, 0))$$

[3]This is the reason why the current setting is not a special case of the combinatorial V setting where the valuation of a set of tasks is the valuation of the subset for which the prerequisites are met.

Instance 2: The true types are θ_1' and θ_2, and the declared types are θ_1, and θ_2'. Thus, the only difference from instance 1 is agent 1's true type which is insignificant, because agent 1 never gets to attempt a task. Thus, we have a similar expected utility function:

$$\bar{u}_1(c_1', \theta_1, \theta_2', (p_1', p_2)) = R_1((\theta_1, \theta_2'), (0, 0))$$

Instance 3: The true types are θ_1' and θ_2, and the declared types are θ_1', and θ_2'. Now we have also changed agent 1's declared type to θ_1'. Both tasks will be allocated to the null agent: $A_1(\theta_1', \theta_2') = A_2(\theta_1', \theta_2') = (0, 0)$. Therefore, $\mu = (0, 0)$ still holds with probability 1, and we get the following equations:

$$\bar{u}_1(c_1', \theta_1', \theta_2', (p_1', p_2)) = R_1((\theta_1', \theta_2'), (0, 0))$$

$$\bar{u}_2(c_2, \theta_2', \theta_1', (p_1', p_2)) = R_2((\theta_1', \theta_2'), (0, 0))$$

If $R_2((\theta_1', \theta_2'), (0, 0)) < 0$, then IR would be violated if θ_2' were indeed the true type of agent 2, because the assignment function would be the same. Since the center thus receives no payment from agent 2, and it never gains any utility from completed tasks, the CR condition requires that $R_1((\theta_1', \theta_2'), (0, 0)) \leq 0$. Thus, $\bar{u}_1(c_1', \theta_1', \theta_2', (p_1', p_2)) \leq 0$.

Notice that if agent 1 lied in this instance and declared her type to be θ_1, then we are in instance 2. So, to preserve IC, agent 1 must not have incentive to make this false declaration. $\bar{u}_1(c_1, \theta_1, \theta_2', (p_1', p_2)) = R_1((\theta_1, \theta_2'), (0, 0)) \leq \bar{u}_1(c_1', \theta_1', \theta_2', (p_1', p_2)) \leq 0$.

Instance 1: Now we return to instance 1. Having shown that $R_1((\theta_1, \theta_2'), (0, 0)) \leq 0$, we know that when agent 1 declares truthfully in this instance, her expected utility will be: $\bar{u}_1(c_1', \theta_1, \theta_2', p) \leq 0$.

We will now show that agent 1 must have a positive expected utility if she falsely declares θ_1''. In this case, both tasks are assigned to agent 1. That is, $A_1(\theta_1'', \theta_2') = (1, 1)$. We know that $R_2((\theta_1'', \theta_2'), (1, 1)) \geq 4$ by IR for agent 1, because if θ_1'' were agent 1's true type, then both tasks would be completed and agent 1 would incur a cost of 4.

We now know that if agent 1 falsely declares θ_1'' in instance 1: $\bar{u}_1(c_1', \theta_1'', \theta_2', p) = R_1((\theta_1'', \theta_2'), (1, 1)) - (c_{11} + c_{12}) \geq 4 - 3 \geq 1$. Thus, agent 1 has incentive to falsely declare θ_1'' in instance 1, violating IC. Thus, we have reached a contradiction and completed the proof of the base case.

Inductive Step: We now prove the inductive step, which consists of two parts: incrementing n and incrementing t. In each case, the inductive hypothesis is that no mechanism satisfies IC, IR, CR, and SE for $n = x$ and $t = y$, where $x, y \geq 2$.

Part 1: For the first case, we must show that no mechanism exists that satisfies IC, IR, CR, and SE for $n = x + 1$ and $t = y$, which we will prove by contradiction. Assume that such a mechanism does exist. There is a one-to-one mapping from every instance in which $n = x$ and $t = y$ to an instance that only differs in the addition of an "extra" agent who truthfully declares her type θ_e. Since such an instance satisfies $n = x + 1$ and $t = y$, our mechanism must satisfy IC, IR, CR, and SE for this instance. Because of SE, this mechanism can never assign the task to the extra agent. Because of IR, this mechanism can never receive a positive payment from the extra agent. Since in each instance the only effect that the extra agent can have on the mechanism is to receive a payment from the center, we can transform this mechanism into one which satisfies IC, IR, CR, and SE for all instances where $n = x$ and $t = y$ by simply removing the revenue function and assignment function for the extra agent, contradicting the inductive hypothesis.

Part 2: For the second case, we need to show that no mechanism can satisfy IC, IR, CR, and SE for $n = x$ and $t = y+1$. We use a similar proof by contradiction, starting from the assumption that such a mechanism does exist. There is a one-to-one mapping from every instance in which $n = x$ and $t = y$ to an instance of $n = x$ and $t = y+1$ through the addition of an "extra" task t_e that is not involved in any dependencies and for which the center has no value for its completion. By SE, if a satisfying mechanism exists, then there exists a satisfying mechanism that always assigns this task to the dummy agent (recall that this is equivalent to not assigning the task). We can transform this mechanism into one which satisfies our goals for $n = x$ and $t = y$ by simply removing the assignment of t_e to the dummy agent. Once again, we have contradicted the inductive hypothesis, and the proof is complete. ■

Interestingly, by slightly altering our mechanism we can solve the problem in an equilibrium.

Equilibrium Multiple Task Mechanism:

Assignment The chosen assignment A maximizes the (declared) expected welfare $\bar{W}(A, \hat{c}, \hat{p}) = E_{[\mu|A,\hat{p}]}[W(A, \hat{c}, \mu)]$.

Payment The payment to each agent i is defined according to her completion vector: $R_i = \bar{W}_{-i}(A, \hat{c}_{-i}, \mu) - \bar{W}^*_{-i}(\hat{c}_{-i}, \hat{p}_{-i})$

The only difference from the Multiple Task Mechanism is that the first term of the payment rule uses the *actual* completion vector, instead of the distribution induced by the declaration of the other agents. As a result, our properties are satisfied only as an equilibrium: if all agents declare truthfully, then no agent has incentive to deviate to a different declaration. While there is no formal name for this type of equilibrium, it is similar in spirit to a Nash equilibrium, but technically different because there is no common knowledge of the game (since privately-known types affect the utility of other agents). It is also similar to a Bayes-Nash equilibrium, but much stronger because it holds regardless of agent beliefs about the prior distributions for the types of the other agents. We define Equilibrium IC to hold if truth-telling is such an equilibrium. Equilibrium IR and SE are defined similarly.

Theorem 6 *The Equilibrium Multiple Task Mechanism satisfies NFR, Equilibrium IC, Equilibrium IR, and Equilibrium SE, even when dependencies exist.*

6 COST VERIFICATION

A practical drawback of our mechanisms is that the payments (or fines) may be very large, especially when service provider is far more efficient than the other agents. Also, since CR is not satisfied in our most general settings, the designer has to take a risk.

Previous work (Nisan & Ronen 2001) has stressed the importance of ex post verification. It showed that when the designer can verify the costs and the actions of the agents *after* the work was done, the power of the designer increases dramatically. All of our previous constructions have corresponding versions that use verification. The main advantage of these mechanisms is that the payments can be *normalized* by any linear function, thus making the potential losses more reasonable for both the agents and the center. Due to space constraints we omit these constructions.

7 DISCUSSION & FUTURE WORK

In this paper we studied task allocation problems in which agents may fail to complete their assigned tasks. For the settings we considered (single task, multiple tasks with combinatorial properties, and multiple tasks with dependencies) we provided either a mechanism that satisfies our goals or an impossibility result.

It is worth pointing out that all of the results in this paper hold when we expand the set of possible failures to include rational, intentional failures, which occur when an agent increases her utility by not attempting an assigned task (and thus not incurring the corresponding cost). Modelling this possibility would complicate our model without changing any of our results. Intuitively, our positive results continue to hold because the payment rule aligns an agent's utility with the welfare of the system. If failing to attempt some subset of the assigned tasks would increase the welfare, then these tasks would not have been assigned to any agent. Obviously, all impossibility results would still hold when we expand the set of possible actions for the agents.

Many interesting directions stem from this work. Two possibilities are retrying tasks after a failure or allowing multiple agents to attempt the same task in parallel. The computation of our allocation and payment rules presents non-trivial algorithmic problems. Also, the payment properties for the center may be further investigated, especially in settings where CR must be sacrificed to satisfy our other goals.

Finally, we believe that the most important future work will be to consider a wider range of possible failures, and to discover new mechanisms to overcome them. In particular, we would like to explore the case in which agents may fail maliciously or irrationally. For this case, even developing a reasonable model of the setting provides a major challenge.

References

Boutilier, C.; Shoham, Y.; and Wellman, M. 1997. Special issue on economic principles of multi-agent systems. *Artificial Intelligence* 94.

Clarke, E. 1971. Multipart pricing of public goods. *Public Choice* 18:19–33.

d'Aspremont, C., and Gerard-Varet, L. 1979. Incentives and incomplete information. *Journal of Public Economics* 11(1):25–45.

Eliaz, K. 2002. Fault tolerant implementation. *Review of Economic Studies.* Forthcoming.

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

Linial, N. 1994. Game theoretic aspects of computing. In *Handbook of Game Theory*, volume 2. Elsevier Science Publishers B.V. 1339–1395.

Mas-Collel, A.; Whinston, W.; and Green, J. 1995. *Microeconomic Theory.* Oxford university press.

Nisan, N., and Ronen, A. 2001. Algorithmic mechanism design. *Games and Economic Behavior, special issue on game theory and AI* 35:166–196.

Parkes, D. C., and Ungar, L. H. 2000. Iterative combinatorial auctions: Theory and practice. In *AAAI,IAAI*, 74–81.

Rosenschein, J. S., and Zlotkin, G. 1994. *Rules of Encounter.* MIT Press.

Shoham, Y., and Tennenholtz, M. 2001. The fair imposition of tasks in multi-agent systems. In *IJCAI*, 1083–1088.

Vickrey, W. 1961. Counterspeculations, auctions, and competitive sealed tenders. *Journal of Finance* 16:15–27.

Bidding Clubs in First-Price Auctions[*]

Kevin Leyton-Brown Yoav Shoham Moshe Tennenholtz

Abstract

We introduce a class of mechanisms, called *bidding clubs*, that allow agents to coordinate their bidding in auctions. Bidding clubs invite a set of agents to join, and each invited agent freely chooses whether to accept the invitation or to participate independently in the auction. Agents who join a bidding club first conduct a "knockout auction" within the club; depending on the outcome of the knockout auction some subset of the members of the club bid in the primary auction in a prescribed way. We model this setting as a Bayesian game, including agents' choices of whether or not to accept a bidding club's invitation. After describing this general setting, we examine the specific case of bidding clubs for first-price auctions. We show the existence of a Bayes-Nash equilibrium where agents choose to participate in bidding clubs when invited and truthfully declare their valuations to the coordinator. Furthermore, we show that the existence of bidding clubs benefits all agents (both inside and outside of bidding clubs).

1 Introduction

Most work on auctions concentrates on the design of auction protocols from the seller's perspective, and in particular on optimal (i.e., revenue maximizing) auction design. In this paper we present a class of systems to assist sets of bidders, *bidding clubs.* The idea is similar to the idea behind "buyer clubs": aggregating the market power of individual bidders. Buyer clubs work when buyers' interests are perfectly aligned; the more buyers join in a purchase the lower the price for everyone. In auctions it is relatively easy for multiple agents to cooperate, hiding behind a single auction participant. Intuitively, these bidders can reduce their payment if they win, by causing others to lower their bids in the case of a first-price auction or by possibly removing the second-highest bidder in the case of a second-price auction. However, the situation in auctions is not as simple as in buyer clubs, because while bidders can gain by sharing information, the competitive nature of auctions means that bidders' interests are not aligned. Thus there is a complex strategic relationship among bidders in a bidding club, and bidding club rules must be designed accordingly.

[*]Thanks to Navin Bhat and Ryan Porter for very helpful discussions about Theorem 3. This work was supported by DARPA grant F30602-00-2-0598 and a Stanford Graduate Fellowship.

1.1 Related Work

Below we discuss the most relevant previous work and its relation to ours, noting the relative scarcity of previous work on bidder-centric mechanisms. This work all comes under the umbrella of self-enforcing collusive protocols for non-repeated auctions. *Collusion* is a negative term reflecting a seller-oriented perspective; since we adopt a more neutral stance towards such bidder activities, we use the term *bidding clubs* rather than the terms *bidding rings* and *cartels* that have been used in the past. However, the technical development is not impacted by such subtle differences in moral attitude.

1.1.1 Collusion in Second-Price Auctions

One of the first formal papers to consider collusion in second-price auctions was written by Graham and Marshall [3]. This paper introduces 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 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 [4]. The case where only some of the agents are part of the cartel is discussed by Mailath and Zemsky [9]. Ungern and Sternberg [14] discuss collusion in second-price auctions where the designated winner of a cartel is not the agent with the highest valuation. Although not presented in any existing work of which we are aware, it is also easy to extend Graham and Marshall's protocol to an environment where multiple cartels may operate in the same auction alongside independent bidders.

1.1.2 Collusion in First-Price Auctions

There is little formal work on collusion in first-price auctions, the most important exception being a very influential paper by McAfee and McMillan [11]. It is the closest in the literature to our work, and indeed we have borrowed some modelling elements from it. Several sections, including the discussion of enforcement and the argument for independent private values as a model of agents' valuations, are directly applicable to our paper. However, the setting introduced in their 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) when non-cartel members bid without taking the existence of a cartel into account and (2) when 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. Finally, a brief presentation of "cartel-formation games" is related to our discussion of agents' decision of whether or not to accept an invitation to join a bidding club.

1.1.3 Other Work on Collusion

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 [6]. A discussion and comparison of the stability of rings associated with classical auctions can be found in [13], 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 (see [2]) and in the context of general Bertrand or Cournot competition (see [1]).

Our previously published work anticipates some of the results reported here. Specifically, in [7] we considered bidding clubs under the assumptions that only a single bidding club exists, and that bidders who were not invited to join the club are not aware of the possibility that a bidding club might exist. The current paper is an extension and generalization of that earlier work. An extended abstract of the current paper appeared in AAAI-02 [8].

2 Technical Preliminaries

Our goal is to extend on past work on bidder cooperation in first-price auctions to the standard game-theoretic setting in which *all* agents (both cartel members and non-members) are rational, and act in equilibrium based on true knowledge of the economic environment. We also want to increase realism by allowing for the possibilities that more than one cartel will exist (introducing the new wrinkle that cartel members must reason about the behavior of other cartels) and that some agents will not belong to any cartel. Of course we also want to allow for real-numbered valuations drawn from an interval, as compared to the case studied in [11] where valuations take one of only two discrete values.

2.1 Auction Setting

In this section we give a formal description of the auction setting and introduce notation. 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 distribution, F, having density function f. 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 will be varied throughout the paper; in E_c let $\mathcal{S} = \{\varnothing\}$. 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 $p_n^{\tau_i}$ we denote the probability that agent i assigns to there being exactly n agents in the auction, conditioned on his type τ_i. We denote the whole distribution conditioned on i's type as uppercase P^{τ_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 it is 0 otherwise. Thus, we assume that there are no externalities in agents' valuations and that agents are risk-neutral. $b_i : \mathcal{T} \to \mathbb{R}$ 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, indicating that i will not participate in the auction.

2.2 Classical first-price auctions

It is instructive to consider the reasons why most previous work in collusion has focused on second-price rather than first-price auctions. Since second-price auctions give rise to dominant strategies, and since colluding agents can gain by having other agents drop out without changing their own bidding behavior, it is possible to study collusion in many settings related to these auctions without performing strategic equilibrium analysis. In particular, agents outside a cartel have no reason to change their strategies if they suspect (or even know) that collusion is taking place. In first-price auctions agents who are not part of the cartel must take into account the likelihood of collusion in deciding what they should bid, since their strategy amounts to predicting the second-highest bid conditional on their bid being highest, and this computation depends on the total number of agents. The settings in [11] are largely designed to overcome this problem: e.g., if all agents belong to the cartel, or if non-cartel agents are assumed to play as though collusion is impossible, the question of how cartel members and non-cartel members reason about each other is avoided.

This suggests that the choice of information structure will make a real difference for the study of collusion in first-price auctions. The most familiar is what we will call the "classical" first-price auction, where the number of participants is part of the economic environment (as in E_c). The equilibrium analysis of classical first-price auctions is quite standard[1]:

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

Proposition 1 *If valuations are selected independently according to the uniform distribution on* $[0, 1]$ *then it is a unique symmetric equilibrium for each agent* i *to follow the strategy:*

$$b(v_i) = \frac{n-1}{n} v_i.$$

Using classical equilibrium analysis (e.g., following Riley and Samuelson [12]) classical first-price auctions can be generalized to an arbitrary continuous distribution F.

Proposition 2 *If valuations are selected from a continuous distribution* F *then it is a unique symmetric equilibrium for each agent* i *to follow the strategy:*

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

.

In both cases, observe that the strategy is parameterized by valuation, and 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:

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

We are interested in constructing a bidding club protocol: a collusive agreement that requires low bidders to drop out of the main auction if they lose in a knockout auction. It is immediately obvious that such collusion is nonsensical in a classical first-price auction. Since all agents have full knowledge of the economic environment, they all know the true number of agents; as a result, it will not matter to agents outside a cartel if cartel members with low valuations drop out, and so the original equilibrium (based on the true number of agents in the environment) will still hold. This seems more of a problem with our auction model than with collusion in first-price auctions *per se*—in practice bidders do not know how many agents have declined to participate, because they don't actually know the number of agents in the economic environment. The next section considers an economic environment that addresses this issue.

2.3 First-price auctions with 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 chosen stochastically from a probability distribution, and while the number of participants is not known to the individual agents (not being part of the economic environment) the *distribution* is commonly known [10]. This setting requires that we redefine the economic environment; denote the new economic environment as E_s. Let the set of agents who may participate in the economic environment be $\mathcal{A} \equiv \mathbb{N}$. Let

β_A represent the probability that a finite set $A \subset \mathcal{A}$ is the set of agents. The probability that n agents will participate in the auction is $\gamma_A(n) = \sum_{A,|A|=n} \beta_A$. All agents know the probability distribution β_A. Once an agent k is selected, he updates his probability of the number of agents present as:

$$p_n^k = \frac{\sum_{A,|A|=n,k\in A} \beta_A}{\sum_{A,k\in A} \beta_A}. \tag{2}$$

We deviate from the model in [10] by adding the assumption that all bidders are equally likely to be chosen. Hence p_n^k is the same for all k; we will hereafter refer only to p_n. Finally, we assume that $\gamma_A(0) = \gamma_A(1) = 0$; at least two agents will participate in the auction.

An equilibrium for this setting was demonstrated by Harstad, Kagel and Levin [5]:

Proposition 3 *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:*

$$b(v_i) = \sum_j \frac{F^{j-1}(v_i)p_j}{\sum_k F^{k-1}(v_i)p_k}\, b^e(v_i, j)$$

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 2.2 above. P is deduced from the economic environment.[2] We overload our previous notation for the equilibrium bid, this time as a function of the agent's valuation and the probability distribution P:

$$b^e(v_i, P) = \sum_j \frac{F^{j-1}(v_i)p_j}{\sum_k F^{k-1}(v_i)p_k}\, b^e(v_i, j) \tag{3}$$

Unfortunately, this auction model is still not rich enough to express our intuition about how agents could collude in a first-price auction. If each agent knows only a distribution on the total number of agents interested in participating in the auction, then he has no way of knowing that other agents have dropped out! It seems reasonable that agents will sometimes know how many agents are *placing bids* in the auction, even though they may not know the number of agents who chose not to participate at all. 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 we could hope collusion based on dropping agents with low valuations would work. We must first introduce a new type of auction to model this auction hall scenario.

[2]Recall that P is a set: $p_j \in P$ for all $j \geq 0$, where p_j denotes the probability that the economic environment contains exactly j agents.

2.4 First-price auctions with participation revelation

First-price auctions with participation revelation are defined 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 4 *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 are always better off participating in first-price auctions as long as there is no participation fee. The only way of participating is to declare the intention to participate in the first phase of the auction. Thus the number of agents announced by the auctioneer is equal to the total number of agents in the economic environment. From proposition 2 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. ■

Settings modelled using classical first-price auctions may often be more appropriately modelled as first-price auctions with participation revelation, since bidders rarely 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, explaining why the simpler classical model is commonly used. For the study of bidding clubs, however, the difference between the mechanisms is profound—we are now able to make the standard assumption that bidders have complete knowledge of the economic environment, while still finding that bidder strategies are affected by the number of other agents who indicate an intention to participate in the auction.

2.5 Distinguishing Features of our Model

Having justified our setting, it is worthwhile to emphasize the main differences between our model of collusion and models proposed in the work surveyed above (particularly [4] and [11]):

1. The number of bidders is stochastic.

2. There is no minimum number of bidders in a bidding club (e.g., bidding clubs are not required to contain all bidders).[3]

3. There is no limit to the number of bidding clubs in a single auction.

4. Club members and independent bidders behave strategically, acting according to correct beliefs about their environment.

Additionally, we make several restrictions on the bidding club protocols that we are willing to consider. None of these is required for the construction of a working protocol, but we feel that each of these characteristics is necessary for bidding clubs to be a realistic model of bidder cooperation:

1. Participation in bidding clubs requires an invitation, but bidders must be free to decline this invitation without (direct) penalty. In this way we include the choice to collude as one of agents' strategic decisions, rather than starting from the assumption that agents will collude.

2. Bidding club coordinators must make money on expectation. This ensures that third-parties have incentive to run bidding club coordinators.

3. The bidding club protocol must give rise to an equilibrium where all invited agents choose to participate, even when the bidding club operates in a single auction as opposed to a sequence of auctions. This means that agents can not be induced to collude in a given auction by the threat of being denied future opportunities to collude.

2.6 Overview

Section 3 expands the auction models and economic environments described above to the bidding club setting. Section 4 examines bidding club protocols for first-price auctions. After giving assumptions and two lemmas, we give a bidding club protocol for first-price auctions with participation revelation. Our main technical results are that:

- It is an equilibrium for agents to accept invitations to join bidding clubs when invited and to disclose their true valuations to their bidding club's coordinator, and for singleton agents to bid as they would in an auction with a stochastic number of participants in an economic environment without bidding clubs, in which the distribution over the number of participants is the same as in the bidding clubs setting.

- In equilibrium each agent is better off as a result of his own club (that is, his expected payoff is higher than would have been the case if his club never existed, but other clubs—if any—still did exist).

[3]For technical reasons we will have to assume that there is a finite *maximum* number of bidders in each bidding club; however, this maximum may be any integer greater than or equal to two.

- In equilibrium each club increases all non-members' expected payoffs, as compared to equilibrium in the case where all club members participated in the auction as singleton bidders, but all other clubs—if any—still existed.
- In equilibrium each agent is either better off or equally well off belonging to a bidding club as compared to equilibrium in the case where no clubs exist.

Finally, section 5 touches on questions of trustworthiness of coordinators, legality of bidding clubs and steps an auctioneer could take to disrupt the operation of bidding clubs.

3 Bidding Club Auction Model

In this section we extend both the economic environment and auction mechanism described above to include the characteristics necessary for a model of bidding clubs. As described above, our aim is not to model a situation where agents' *decision* to collude is 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 colluding agents must take into account the possibility that *other* groups of agents in the auction may also be colluding. Once we have defined the new economic environment and auction mechanism, a well-defined Bayesian game will be specified by every tuple of primary auction type, bidding club rules and distributions of agent types, number of agents and number of bidding clubs.

3.1 The Economic Environment

We extend the economic environment E_s to consist of a set of agents who have non-negative valuations for a good at auction, the distinguished agent 0 and a set of bidding club coordinators who do not value the good, but may invite agents to participate in a bidding club. We will denote the new economic environment E_{bc}. Intuitively, in E_{bc} an agent's belief update after observing the number of agents in his bidding club does not result in any change in the distribution over the number of *other* agents in the auction, because the number of agents in each bidding club is independent of the number of agents in every other bidding club.

3.1.1 Coordinators

Coordinators 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. We select coordinators in a process analogous to the approach for exogenously selecting agents in [10]: we draw a finite set of individuals from an infinite set of potential coordinators. In this case, however, this finite set is considered "potential coordinators"; in section 3.1.2 we will describe which potential coordinators are "actualized", i.e., correspond to actual coordinators.

Let $\mathcal{C} \equiv \mathbb{N}$ (excluding 0) be the set of all coordinators. β_C represents the probability that a finite set $C \subset \mathcal{C}$ is selected to be the set of potential coordinators. We add the restriction that all coordinators are equally likely to be chosen. A consequence of this restriction is that an agent's knowledge of the coordinator with whom he is associated does not give him additional information about what other coordinators may have been selected. We denote the probability that an auction will involve n_c potential coordinators as $\gamma_C(n_c) = \sum_{C,|C|=n_c} \beta_C$. We assume that $\gamma_C(0) = \gamma_C(1) = 0$: at least two potential coordinators will be associated with each auction.

3.1.2 Agents

We independently associate a random number of agents with each potential coordinator, again drawing a finite set of actual agents from an infinite set of potential agents. If only one (actual) agent is associated with a potential coordinator, the potential coordinator will not be actualized and hence the agent will not belong to a bidding club. In this way we model agents who participate directly in the auction without being associated with a coordinator. If more than one agent is associated with a potential coordinator, the coordinator *is* actualized and all its associated agents receive an invitation to participate in the bidding club.

Let $\mathcal{A} \equiv \mathbb{N}$ be the set of all agents, and let $\kappa \in \mathbb{N} \setminus \{0, 1\}$ be the maximum number of agents who may be associated with a single bidding club. Partition $\mathcal{A}$ into subsets, where agent i belongs to the subset $\mathcal{A}_{\lceil i/\kappa \rceil}$. Let β_A be the probability that a finite set $A \subset \mathcal{A}_i$ is the set of agents associated with potential coordinator i; we assume that all agents are equally likely to be chosen. The probability that n agents will be associated with a potential coordinator is denoted $\gamma_A(n) = \sum_{A,|A|=n} \beta_A$. By the definition of κ, $\forall j > \kappa, \gamma_A(j) = 0$; we assume that $\gamma_A(0) = 0$ and that $\gamma_A(1) < 1$.

3.1.3 Types and Signals

Recall that the type $\tau_i \in \mathcal{T}$ of agent i is the pair $(v_i, s_i) \in V \times \mathcal{S}$. Let $\mathcal{S} \in \mathbb{N} \setminus \{0\}$; s_i denotes agent i's private information about the number of agents in his bidding club.[4] Of course, if this number is 1 then there is no coordinator for the agent to deal with, and he will simply participate in the main auction. Note also that agents are neither aware of the number of potential coordinators

[4]In fact, none of our results require that agents know the *number* of agents in their bidding clubs; it would be sufficient that agents know *whether* they belong to a bidding club. We consider the setting where agents' signals are more informative because deviation from the bidding club protocol is *more* profitable in this case.

for their auction nor the number of actualized potential coordinators, though they are aware of both distributions.

3.1.4 Beliefs

Once an agent is selected, he updates his probability distribution over the number of actual agents in the economic environment. Not all agents will have the same beliefs—agents who have been signaled that they belong to a bidding club will expect a larger number of agents than singleton agents. We denote by $p_m^{n,k}$ the probability that there are a total of m agents in the auction, given that there are n bidding clubs and that there are k agents in the bidder's own club; we denote the whole distribution $P^{n,k}$. Because the numbers of agents in each bidding club are independent, observe that every agent in the whole auction has the same beliefs about the number of other agents in the economic environment, discounting those agents in his own bidding club. Hence agent i's beliefs are described by the distribution P^{n,s_i}.

3.2 The Augmented Auction Mechanism

Bidding clubs, in combination with a main auction, induce an augmented auction mechanism for their members:

1. A set A of bidders is invited to join the bidding club.

2. Each agent i sends a message μ_i to the bidding club coordinator. This may be the null message, which indicates that i will not participate in the coordination and will instead participate freely in the main auction. Otherwise, i agrees to be bound by the bidding club rules, and μ_i is i's declared valuation for the good. Of course, i can lie about his valuation.

3. Based on commonly-known rules and the information all the members supply, the coordinator selects a subset of the agents to bid in the main auction. We assume that the coordinator can force agents to bid as desired, e.g. by imposing a punitive charge on misbehaving agents.

4. The coordinator makes a payment to each club member. The amount of the payment must not depend on any of the agents' declared valuations or on the outcome of the main auction.

5. If a bidder represented by the coordinator wins the main auction, he is made to pay the amount required by the auction mechanism to the auctioneer. In addition, he may be required to make an additional payment to the coordinator.

Any number of coordinators may participate in an auction. However, we assume that there is only a single coordination protocol, and that this protocol is common knowledge.

4 Bidding Clubs for First-Price Auctions

This section contains the paper's main technical results. We begin by stating some (mild) assumptions about the distribution of agent valuations, then use these assumptions to prove a technical lemma. A second lemma explains how we can show the existence of an equilibrium in a setting where agents receive asymmetric information and are subject to asymmetric payment rules. We then give the bidding club protocol for first-price auctions, based on a first-price auction with participation revelation as described in section 2.4. We show an equilibrium of this auction, and demonstrate that agents gain under this equilibrium.

4.1 Assumptions about F

Our results hold for a broad class of distributions of agent valuations—all distributions for which the following two assumptions are true.

Assumption 1 *F is continuous and atomless.*

In order to give our second assumption, we must introduce some notation:

$$P_{x\geq i} = \sum_{x=i}^{\infty} p_x. \tag{4}$$

We now define the relation "<" for probability distributions:

$$P < P' \text{ iff } \exists l (\forall i < l, P_{x\geq i} = p'_{x\geq i} \text{ and } \forall i \geq l, P_{x\geq i} < P'_{x\geq i}). \tag{5}$$

We are now able to state our second assumption:

Assumption 2 $(P < P')$ *implies that* $\forall v$, $b^e(v, P) < b^e(v, P')$

Intuitively, we assume that every agent's symmetric equilibrium bid in E_s with number of participants drawn from P' is strictly greater than that agent's symmetric equilibrium bid in E_s with number of participants drawn from P, in the case where P' stochastically dominates P.[5]

4.2 A Technical Lemma

It is important to note that the notation $P^{n,k}$ may be seen as defining a probability distribution over the number of agents in economic environment E_s (i.e., even without the existence of bidding clubs). It is thus possible to discuss equilibrium bids in the classical stochastic settings where the number of bidders is drawn from such a distribution. While it will remain to show why these values are meaningful in our setting where (among other differences) agents have

[5]This assumption holds for every standard distribution of independent valuations of which we are aware.

asymmetric information, it will be useful to prove the following lemma about the classical stochastic setting:[6]

Lemma 1 $\forall k \geq 2, \forall n \geq 2, \forall v,\ b^e(v, P^{n+k-1,1}) > b^e(v, P^{n,k})$

Remark. This lemma asserts that the symmetric equilibrium bid is always higher when more agents belong to the main auction as singleton bidders and the total number of agents is held constant.

Proof. Recall Assumption 2 from section 4.1. We defined $P < P'$ as the proposition that $\exists l (\forall i < l, P_{x\geq i} = P'_{x\geq i}$ and $\forall i \geq l, P_{x\geq i} < P'_{x\geq i})$, and assumed that $(P < P')$ implies that $\forall v, b^e(v, P) < b^e(v, P')$. It is thus sufficient to show that $P^{n+k-1,1} > P^{n,k}$. We will take $l = n + k$.

First we will show that $\forall j < n + k, P^{n+k-1,1}_{x\geq j} = P^{n,k}_{x\geq j}$. The distribution $P^{n+k-1,1}$ expresses the belief that there are $n+k-2$ potential coordinators, the membership of which is distributed as described in section 3.1, and one potential coordinator that is known to contain only a single bidder. The distribution $P^{n,k}$ expresses the belief that there are $n-1$ potential coordinators, the membership of which is again distributed as described in section 3.1, and one potential coordinator that is known to contain exactly k bidders. Under both distributions it is certain that there are at least $n + k - 1$ agents. Therefore $\forall j < n + k, P^{n+k-1,1}_{x\geq j} = P^{n,k}_{x\geq j} = 1$.

Second, $\forall j \geq n + k$, $P^{n+k-1,1}_{x\geq j} > P^{n,k}_{x\geq j}$. Considering $P^{n+k-1,1}$, observe that for $n + k - 2$ of the potential coordinators the probability that this coordinator contains a single agent is less than one and these probabilities are all independent; the last potential coordinator contains a single agent with probability one. Considering $P^{n,k}$, there are $n - 1$ potential coordinators where the probability of containing a single agent is less than one, exactly as above, and k potential coordinators certain to contain exactly one agent. Thus the two distributions agree exactly about $n - 1$ of the potential coordinators, which both hold to contain more than a single agent, and likewise both distributions agree that one of the potential coordinators contains exactly one agent. However, there remain $k - 1$ potential coordinators about which the distributions disagree; $P^{n+k-1,1}$ always generates a greater or equal number of agents for these potential coordinators, as compared to $P^{n,k}$. Under the latter distribution all these agents are singletons with probability one, while under the former there is positive probability that each of the potential coordinators contains more than one agent. As long as $k \geq 2$, there is at least one potential coordinator for which $P^{n+k-1,1}$ stochastically dominates $P^{n,k}$. Thus $\forall k \geq 2, \forall n \geq 2, \forall v\, P^{n+k-1,1} > P^{n,k}$. ■

[6]For convenience and to preserve intuition in what follows we will refer to the number of potential coordinators and the number of agents belonging to a coordinator even though we concern ourselves with the economic environment E_s where bidding clubs do not exist. The number of potential coordinators is shorthand for the number n_c drawn from γ_C in the first phase of the procedural definition of the distribution $P^{n,k}$. Likewise the number of agents associated with a potential coordinator is shorthand for the number of agents chosen from one of the n_c iterative draws from γ_A.

4.3 Truthful Equilibria in Asymmetric Mechanisms

In E_{bc} there is informational asymmetry because agents receive different signals, and asymmetric payment rules because some agents belong to bidding clubs of different sizes and others do not belong to a bidding club at all. The lemma in this section will allow us to go on to show an equilibrium in Theorem 1 despite these asymmetries.

We describe a particular class of auction mechanisms that are asymmetric in the sense that every agent is subject to the same allocation rule but to a potentially different payment rule, and furthermore that agents may receive different signals. A truth-revealing equilibrium exists in such auctions when the following conditions hold:

1. The auction allocates the good to the agent who submits the highest bid.
2. Consider the auction M_i in which *all* agents are subject to agent i's payment rule and the above allocation rule, and where (hypothetically) *all* agents receive the signal s_i.[7] Truth-revelation is a symmetric equilibrium in M_i.

Observe that the second condition above is less restrictive than it may appear. From the revelation principle we can see that for every auction with a symmetric equilibrium there is a corresponding auction in which truth-revealing is an equilibrium that gives rise to the same allocation and the same payments for all agents. M_i can thus be seen as a revelation mechanism for any other auction that has a symmetric equilibrium.

Definition 1 *$\bar{M}$ is a* regular asymmetric *auction if it has the following structure, where $\boldsymbol{M}$ represents a set of auctions $\{M_1, \ldots, M_n\}$ which each allocate the good to the agent who submits the highest bid, and which are all truth-revealing direct mechanisms for n risk-neutral agents with independent private valuations drawn from the same distribution:*

1. *Each agent i sends a message μ_i to the center.*
2. *The center allocates the good to the agent i with $\mu_i \in \max_j \mu_j$. If multiple agents submit the highest message, the tie is broken in some arbitrary way.*
3. *Agent i is made to transfer $t_i(\mu, \pi)$ to the center. The transfer function t_i is taken from $M_i \in \boldsymbol{M}$.*

Lemma 2 *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_i, and all agents' strategies. Assume that the other agents are truth revealing, then the other agents' behavior, the allocation rule, and

[7]That is, for every agent j in the real auction, we create an agent k in the hypothetical auction M_i having type $\tau_k = (v_j, s_i)$.

agent i's payment rule are all identical in $\bar{M}$ and M_i. Since truth-revelation is an equilibrium in M_i, truth-revelation is agent i's best response in $\bar{M}$. ■

The next corollary, following directly from Lemma 2, compares a single agent's expected utility under two different auctions which implement different payment rules. We will need this result for our proof of Theorem 1.

Corollary 1 *Consider two regular asymmetric auctions $\bar{M}$ and $\bar{M}'$, which both implement the same transfer function for agent i. In equilibrium, agent i's expected utility is the same in both $\bar{M}$ and $\bar{M}'$.*

Proof. The payoff of agent i is uniquely determined by the allocation rule, its transfer function, and all agents' strategies. Both $\bar{M}$ and $\bar{M}'$ have the same allocation rule. Lemma 2 tells us that truth revelation is a best response for all agents in both $\bar{M}$ and $\bar{M}'$, so all agents' strategies are identical in the two auctions. In general, agents may not receive the same expected utility from $\bar{M}$ and $\bar{M}'$. However, since i has the same transfer function in both auctions, i's expected utility in $\bar{M}$ is equal to his expected utility in $\bar{M}'$. ■

4.4 First-Price Auction Bidding Club Protocol

What follows is the protocol of a coordinator who approaches k agents.

1. Each agent i sends a message μ_i to the coordinator.
2. If at least one agent declines participation then the coordinator registers in the main auction for every agent who accepted the invitation to the bidding club. For each bidder i, the coordinator submits a bid of $b^e(\mu_i, P^{n,k})$, where n is the number of bidders announced by the auctioneer.
3. If all k agents accepted the invitation then the coordinator drops all bidders except the bidder with the highest reported valuation, who we will denote as bidder h. For this bidder the coordinator places a bid of $b^e(\mu_h, P^{n,1})$ in the main auction.
4. The coordinator pays each member a pre-determined payment $c \geq 0$ whenever all bidders participate in the club, and regardless of the outcome of the auction and of how much each bidder bid. Following the argument in [3] let g be the coordinator's *ex ante* expected gain if all agents behave according to the equilibrium in Theorem 1; the coordinator will not lose money on expectation if it pays each agent $c = \frac{1}{k}(g - c')$ with $0 \leq c' \leq g$.
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 coordinator.

Observe that in equilibrium the coordinator has an expected profit of c', though it will lose kc whenever the winner of the main auction does not belong to its club. If a coordinator wanted to be budget-balanced on expectation rather than profitable on expectation, it could set $c' = 0$.

We are now ready to prove the main theorem of the paper:

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

Proof. We first prove that the above strategy is in equilibrium for both categories of bidders assuming that agents all participate; we then prove that participation is rational for all agents.

For the proof of equilibrium we consider a one-stage mechanism which behaves as follows:

1. The center announces n, the number of bidders in the main auction.
2. Bidders submit bids (messages) to the mechanism.
3. The bidder with the highest bid is allocated the good.
4. The winning bidder is made to pay $b^e(v_i, P^{n,s_i}) - c$.
5. All non-winning bidding club members are paid c.

This one-stage mechanism has the same payment rule for bidding club bidders as the bidding club protocol given above, but no longer implements a first-price payment rule for singleton bidders. In order to prove that the strategies given in the statement of the theorem are an equilibrium, it is sufficient to show that truthful bidding is an equilibrium for all bidders under the given one-stage mechanism. Observe that this mechanism may be seen as a mechanism $\bar{M}$ in the sense of Lemma 2: it allocates the good to the agent who submits the highest message, and (by definition of b^e) the auction M_i in which *all* agents are subject to agent i's payment rule and receive the signal s_i has truth revelation as a symmetric equilibrium.

Strategy of non-club bidder: Assume that all bidding club agents (if any) bid truthfully. Further assume that all non-club agents also bid truthfully except for non-club bidder i. The probability distribution $P^{n,1}$ correctly describes the distribution of the number of agents faced by i, given his signal $s_i = 1$ and the auctioneer's announcement that there are n bidders in the main auction. Although agents in bidding clubs have additional information about the number of agents—each agent knows that there is at least one other agent in his own club—their prescribed behavior is to place bids of $b^e(\mu, P^{n,1})$ in the main auction. Agent i thus faces an unknown number of agents distributed according to $P^{n,1}$ and all bidding $b^e(v, P^{n,1})$. The auction is regular asymmetric: using the result from Lemma 2, i's strategic decision is the same as under a mechanism where all agents are subject to his payment rule and share his signal s_i, and with a stochastic number of bidders distributed according to $P^{n,1}$. In particular, it does not matter that the club members are subject to different payment rules and have additional information, and so i will also bid $b^e(v, P^{n,1})$.

Strategy of club bidder: Assume that all agents accept the invitation to join their respective clubs and then truthfully declare their valuations, excluding club bidder i who decides to participate but considers his bid. Once again, observe that the auction is regular asymmetric, and so Lemma 2 applies: $P^{n,k}$ describes the distribution over the number of agents conditioned on i's signal $s_i = k$, and the bidder submitting the highest (global) message will always be allocated the good. Therefore truthful bidding is a best response for agent i, despite the information asymmetry. Because i gets the payment c regardless of the amount of his bid, the presence or absence of this payment has no effect on his choice of what amount to bid given the decision to participate.

We now turn to the question of participation; for this part of the proof we consider the original, multi-stage mechanism.

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

Participation of club bidder: Assume that $c = 0$; clearly $c > 0$ only increases agents' incentive to participate in a bidding club. Because there is no participation fee, all bidding club bidders will participate in the auction, but must decide whether or not to accept their coordinators' invitations. Assume that all agents except for i join their respective clubs and bid truthfully, and agent i must decide whether or not to join his bidding club. Agent i knows the number of agents in his bidding club and updates his distribution over the number of agents in the whole auction as $P^{n,k}$.

Consider the classical stochastic case where all bidders have the same information as i (and are subject to the same payment rules): from proposition 3 it is a best response for i to bid $b^e(v_i, P^{n,k})$. In this setting i's expected gain is the same as in the equilibrium of the one-stage mechanism from the first part of the proof where all bidding club members (including i) join their clubs and bid truthfully (with $c = 0$), by Corollary 1.

As a result of i declining the offer to participate in the bidding club there are $n - 1$ bidders in the main auction placing bids of $b^e(v, P^{n+k-1,1})$ and $k - 1$ other bidders placing bids of $b^e(v, P^{n,k})$. We know from Lemma 1 that $b^e(v, P^{n+k-1,1}) > b^e(v, P^{n,k})$. Thus the singleton bidders and other bidding clubs will bid a higher function[8] of their valuations than the bidders from the disbanded bidding club. It always reduces a bidder's expected gain in a first-price auction to cause other bidders to bid above the equilibrium, because it reduces the chance that he will win without affecting his payment if he does win. This is exactly the effect of i declining the offer to join his bidding club: the $k - 1$ other bidders from i's bidding club bid according to the equilibrium of the classical stochastic case discussed above, but the $n - 1$ singleton and bidding club bidders submit bids that exceed the symmetric equilibrium amount. Therefore i's expected gain is smaller if he declines the offer to participate than if he accepts it. ■

[8]Note that this occurs because the singleton bidders and other bidding clubs 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 club might each be independent bidding clubs.

4.5 Do bidding clubs cause agents to gain?

All things being equal, bidders are better off being invited to a bidding club than being sent to the auction as singleton bidders. Intuitively, an agent gains by not having to consider the possibility that other bidders who would otherwise have belonged to his bidding club might themselves be bidding clubs.

Theorem 2 *An agent i has higher expected utility in a bidding club of size k bidding as described in Theorem 1 than he does if the bidding club does not exist and k additional agents (including i) participate directly in the main auction as singleton bidders, again bidding as described in Theorem 1, for $c \geq 0$.*

Proof. Consider the counterfactual case where agent i's bidding club does not exist, and all the members of this bidding club are replaced by singleton bidders in the main auction. We will show that i is better off as a member of the bidding club (even when $c = 0$) than in this case. If there were n potential coordinators in the original auction and k agents in i's bidding club, then the auctioneer would announce $n+k-1$ as the number of participants in the new auction. Under the equilibrium from Theorem 1, as a singleton bidder i will bid $b^e(v_i, P^{n+k-1,1})$. If he belonged to the bidding club and followed the same equilibrium i would bid $b^e(v_i, P^{n,k})$. 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. Lemma 1 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. ■

We can also show that singleton bidders and members of other bidding clubs benefit from the existence of each bidding club in the same sense. Following an argument similar to the one in Theorem 2, other bidders gain from not having to consider the possibility that additional bidders might represent bidding clubs. Paradoxically, as long as $c' > 0$, other bidders' gain from the existence of a given bidding club is greater than the gain of that club's members.

Corollary 2 *In the equilibrium described in Theorem 1, singleton bidders and members of other bidding clubs have higher expected utility when other agents participate in a given bidding club of size $k \geq 2$, as compared to a case where k additional agents participate directly in the main auction as singleton bidders.*

Proof. Consider a singleton bidder in the first case, where the club of k agents does exist. (It is sufficient to consider a singleton bidder, since other bidding clubs bid in the same way as singleton bidders.) Following the equilibrium from Theorem 1 this agent would submit the bid $b^e(v_i, P^{n,1})$. Theorem 2 shows that it is better to belong to a bidding club (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 $b^e(v_i, P^{n+k-1,1})$. Since the distribution $P^{n,k}$ is just $P^{n,1}$ with $k-1$ singleton agents added, $\forall k \geq 2, b^e(v_i, P^{n,1}) < b^e(v_i, P^{n,k})$. Thus $\forall k \geq 2, b^e(v_i, P^{n,1}) < b^e(v_i, P^{n+k-1,1})$. ■

Finally, we can show that agents prefer participating in E_{bc} in the equilibrium from Theorem 1 in a bidding club of size k (thus, where the number of agents is distributed according to $P^{n,k}$) to participating in E_s with number of bidders distributed according to $P^{n,k}$, as long as $c > 0$.

Theorem 3 *For all $\tau_i \in \mathcal{T}$, for all $k \geq 2$, for all $n \geq 2$, for all $c > 0$, agent i obtains smaller expected utility by:*

1. *participating in a first-price auction with participation revelation in E_s with number of bidders distributed according to $P^{n,k}$; than by*
2. *participating in a bidding club of size k in E_{bc} and following the equilibrium from Theorem 1.*

When $c = 0$, agent i obtains the same expected utility in both cases.

Proof. For any efficient auction, an agent i's expected utility EU_i is $\sum_j P_j F^{j-1}(V_i)b$, 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). From proposition 4 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 is:

$$\begin{aligned} EU_{i,pr} =& \sum_j p_j^{n,k} F^{j-1}(v_i) b^e(v_i, j) \qquad (6) \\ =& \frac{\sum_\ell p_\ell^{n,k} F^{\ell-1}(v_i)}{\sum_{\ell'} p_{\ell'}^{n,k} F^{\ell'-1}(v_i)} \sum_j p_j^{n,k} F^{j-1}(v_i) b^e(v_i, j) \\ =& \sum_\ell p_\ell^{n,k} F^{\ell-1}(v_i) \left(\sum_j \frac{p_j^{n,k} F^{j-1}(v_i)}{\sum_{\ell'} p_{\ell'}^{n,k} F^{\ell'-1}(v_i)} b^e(v_i, j) \right) \\ =& \sum_\ell p_\ell^{n,k} F^{\ell-1}(v_i) b^e(v_i, P^{n,k}) = EU_{i,s} \qquad (7) \end{aligned}$$

Equation (7) is agent i's expected utility in a first-price auction with a stochastic number of participants, which we shall denote $EU_{i,s}$. Observe that we make use of the definition of $b^e(v_i, P)$ from equation (3).

We now consider case (2). Let $EU_{i,bc}$ denote agent i's expected utility in E_{bc} as a member of a bidding club of size k, in the equilibrium from Theorem 1. Recall that in this equilibrium the bidder with the globally highest valuation always wins, and that all agents in bidding clubs of size k bid $b^e(v_i, P^{n,k})$ and receive a positive payment of c, which does not depend on the amount of their bids or on whether any agent in the club wins the auction.

$$EU_{i,bc} = \sum_j p_j^{n,k} F^{j-1}(v_i) b^e(v_i, P^{n,k}) + c \qquad (8)$$

Intersecting equations (7) and (8), we get:

$$EU_{i,bc} - EU_{i,pr} = c \tag{9}$$

When $c > 0$, agent i's expected utility is strictly greater in case (2) than in case (1); when $c = 0$ he has the same expected utility in both cases. ■

What about agents who do not belong to bidding clubs? We can show in the same way that they are not harmed by the existence of bidding clubs: they are neither better nor worse off in the bidding club economic environment than facing the same distribution of opponents in a first-price auction with participation revelation.

Corollary 3 *For all $\tau_i \in \mathcal{T}$, for all $n \geq 2$, agent i obtains the same expected utility by:*

1. *participating in a first-price auction with participation revelation in E_s with number of bidders distributed according to $P^{n,1}$; as by*
2. *participating as a singleton bidder in E_{bc} and following the equilibrium from Theorem 1.*

Proof. We follow the same argument as in Theorem 3, except that $k = 1$ and $EU_{i,bc}$ does not include c. Thus we get $EU_{i,bc} = EU_{i,pr}$. ■

5 Discussion

In this section we consider the trustworthiness and legality of coordinators, and also discuss two ways for auctioneers to disrupt bidding clubs in their auctions.

5.1 Trust

Why would a bidding club coordinator be willing to provide reliable service, and likewise why would bidders have reason to trust a coordinator? For example, a malicious coordination protocol could be used simply to drop all its members from the auction and reduce competition. While this is a reasonable concern, our coordinators make a profit on expectation, thus providing incentive for a trusted third party to run a reliable coordination service. Indeed, coordinators would be very inexpensive to run: as their behavior is entirely deterministic, they could operate without any human supervision. The establishment of trust is exogenous to our model; we have simply assumed that all agents trust coordinators and that all coordinators are honest.

5.2 Legality

We have often been asked about the legal issues surrounding the use of bidding clubs. While this is an interesting and pertinent question, it exceeds both our expertise and the scope of this paper. We should note, however, that uses of bidding clubs exist that might not fall under the legal definition of collusion. For example, a corporation could use a bidding club to choose one of its departments to bid in an external auction. In this way the corporation could be sure to avoid bidding against itself in the external auction while avoiding dictatorship and respecting each department's self-interest. Coordinators may also be permitted by the auctioneer: e.g., by an internet market seeking to attract more bidders to its site.

5.3 Disrupting Bidding Clubs

There are two things an auctioneer can do to disrupt bidding clubs in a first-price auction. First, she can permit "false-name bidding." (Our auction model has assumed that each agent may place only a single bid in the auction, and that the center has a way of uniquely identifying agents.) Second, she can refrain from publicly disclosing the winner of the auction.

If bidders can bid both in their bidding clubs and in the main auction, they are better off deviating from the equilibrium in Theorem 1 in the following way. A bidder i can accept the invitation to join the bidding club but place a very low bid with the coordinator; 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 1 because accepting the invitation to join the bidding club ensures that the club 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 coordination protocol. If the bidding club 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 club did not exist or where it was disbanded. However, if false-name bidding is impossible and the winner of the auction is publicly disclosed then the bidding club coordinator can detect an agent who has deviated in this way. Because the agent has agreed to participate in the bidding club the coordinator has the power to punish this agent and make the deviation unprofitable. If either or both of these requirements does not hold, however, the coordinator will be unable to detect defection and so the equilibrium from Theorem 1 will not hold.

6 Conclusion

We have presented a formal model of bidding clubs which in many ways extends models traditionally used in the study of collusion; most importantly, all agents behave strategically based on correct information about the economic environment, including the possibility that other agents will collude. Other features of our setting include a stochastic number of agents and of bidding clubs 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 club is part of an agent's choice of strategy. Bidding clubs make money on expectation, and can optionally be configured so they never lose money. We have showed a bidding club protocol for first-price auctions that leads to a (globally) efficient allocation in equilibrium, and which does not make use of side-payments in the case of $c = 0$. There are three ways of asking the question of whether agents gain by participating in bidding clubs in first-price auctions:

1. Could any agent gain by deviating from the protocol?
2. Would any agent be better off if his bidding club did not exist?
3. Would any agent would be better off in an economic environment that did not include bidding clubs 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 $c > 0$.) We have also shown that each bidding club causes *non-members* to gain in the second sense, and does not hurt them in the third sense. Finally, we have discussed ways for an auctioneer to set up the rules of her auction so as to disrupt the operation of bidding clubs.

References

[1] P.C. Cramton and T.R. Palfrey. Cartel enforcement with uncertainty about costs. *International Economic Review*, 31(1):17–47, 1990.

[2] J.S. Feinstein, M.K. Block, and F.C. Nold. Assymetric behavior and collusive behavior in auction markets. *The American Economic Review*, 75(3):441–460, 1985.

[3] D.A. Graham and R.C. Marshall. Collusive bidder behavior at single-object second-price and english auctions. *Journal of Political Economy*, 95:579–599, 1987.

[4] D.A. Graham, R.C. Marshall, and Jean-Francois Richard. Differential payments within a bidder coalition and the shapley value. *The American Economic Review*, 80(3):493–510, 1990.

[5] R.M. Harstad, J. Kagel, and D. Levin. Equilibrium bid functions for auctions with an uncertain number of bidders. *Economic Letters*, 33(1):35–40, 1990.

[6] K. Hendricks and R.H. Porter. Collusion in auctions. *Annale's D'economie de Statistique*, 15/16:216–229, 1989.

[7] K. Leyton-Brown, Y. Shoham, and M. Tennenholtz. Bidding clubs: institutionalized collusion in auctions. In *ACM Conference on Electronic Commerce*, 2000.

[8] K. Leyton-Brown, Y. Shoham, and M. Tennenholtz. Bidding clubs in first-price auctions. In *The 19th national conference on artificial intelligence*, 2002.

[9] G.J. Mailath and P. Zemsky. Collusion in second-price auctions with heterogeneous bidders. *Games and Economic Behavior*, 3:467–486, 1991.

[10] R.P. McAfee and J. McMillan. Auctions with a stochastic number of bidders. *Journal of Economic Theory*, 43:1–19, 1987.

[11] R.P. McAfee and J. McMillan. Bidding rings. *The American Economic Theory*, 82:579–599, 1992.

[12] J.G. Riley and W.F. Samuelson. Optimal auctions. *American Economic Review*, 71:381–392, 1981.

[13] M.S. Robinson. Collusion and the choice of auction. *Rand Journal of Economics*, 16(1):141–145, 1985.

[14] Thomas von Ungern-Sternberg. Cartel stability in sealed bid second price auctions. *The Journal of Industrial Economics*, 18(3):351–358, 1988.

Mechanism Design for Online Real-Time Scheduling

Ryan Porter*
Computer Science Department
Stanford University
Stanford, CA 94305
rwporter@stanford.edu

April 16, 2004

Abstract

For the problem of online real-time scheduling of jobs on a single processor, previous work presents matching upper and lower bounds on the competitive ratio that can be achieved by a deterministic algorithm. However, these results only apply to the non-strategic setting in which the jobs are released directly to the algorithm. Motivated by emerging areas such as grid computing, we instead consider this problem in an economic setting, in which each job is released to a separate, self-interested agent. The agent can then delay releasing the job to the algorithm, inflate its length, and declare an arbitrary value and deadline for the job, while the center determines not only the schedule, but the payment of each agent. For the resulting mechanism design problem (in which we also slightly strengthen an assumption from the non-strategic setting), we present a mechanism that addresses each incentive issue, while only increasing the competitive ratio by one. We then show a matching lower bound for deterministic mechanisms that never pay the agents.

1 Introduction

We consider the problem of online scheduling of jobs on a single processor. Each job is characterized by a release time, a deadline, a processing time, and a value for successful completion by its deadline. The objective is to maximize the sum of the values of the jobs completed by their respective deadlines. The key challenge in this online setting is that the schedule must be constructed in real-time, even though nothing is known about a job until its release time.

Competitive analysis [5, 9], with its roots in [11], is a well-studied approach for analyzing online algorithms by comparing them against the optimal offline algorithm, which has full knowledge of the input at the beginning of its execution. One interpretation of this approach is as a game between the designer of the online algorithm and an adversary. First, the designer selects the online algorithm. Then, the adversary observes the algorithm and selects the sequence of jobs that maximizes the competitive ratio: the ratio of the value of the jobs completed by an optimal offline algorithm to the value of those completed by the online algorithm.

Two papers paint a complete picture in terms of competitive analysis for this setting, in which the algorithm is assumed to know k, the maximum ratio between the value densities (value divided by processing time) of any two jobs. For $k = 1$, [3] presents a 4-competitive algorithm, and proves that this is a lower bound on the competitive ratio for deterministic algorithms. The same paper

*The author would like to acknowledge Yoav Shoham for his influential comments on drafts of the paper, and Yossi Azar for a useful discussion. This work was supported in part by DARPA grant F30602-00-2-0598.

also generalizes the lower bound to $(1+\sqrt{k})^2$ for any $k \geq 1$, and [14] then presents a matching $(1+\sqrt{k})^2$-competitive algorithm.

The setting addressed by these papers is completely non-strategic, and the algorithm is assumed to always know the true characteristics of each job upon its release. However, in domains such as grid computing (see, for example, [6, 7]) this assumption is invalid, because buyers of processor time choose when and how to submit their jobs. Furthermore, sellers not only schedule jobs but also determine the amount that they charge buyers, an issue not addressed in the non-strategic setting.

Thus, we consider an extension of the setting in which each job is owned by a separate, self-interested agent. Instead of being released to the algorithm, each job is now released only to its owning agent. Each agent now has four different ways in which it can manipulate the algorithm: it decides when to submit the job to the algorithm after the true release time, it can artificially inflate the length of the job, and it can declare an arbitrary value and deadline for the job. Because the agents are self-interested, they will choose to manipulate the algorithm if doing so will cause their job to be completed; and, indeed, one can find examples in which agents have incentive to manipulate the algorithms presented in [3] and [14].

The addition of self-interested agents moves the problem from the area of algorithm design to that of mechanism design. In this setting, a mechanism will take as input a job from each agent, and return a schedule for the jobs and a payment to be made by each agent to the center. The mechanism design goal of incentive compatibility requires that it is always in each agent's best interests to immediately submit its job upon release, and to truthfully declare its value, length, and deadline.

In order to evaluate a mechanism using competitive analysis, the adversary model must be updated. In the new model, the adversary still determines the sequence of jobs, but it is the self-interested agents who determine the observed input of the mechanism. Thus, in order to achieve a competitive ratio of c, an online mechanism must both be incentive compatible, and always achieve at least $\frac{1}{c}$ of the value that the optimal offline mechanism achieves on the same sequence of jobs.

The rest of the paper is structured as follows. In Section 2, we formally define and review results from the original, non-strategic setting. After introducing the incentive issues through an example, we formalize the mechanism design setting in Section 3. In Section 4 we present our first main result, a $((1+\sqrt{k})^2+1)$-competitive mechanism. We also show how we can simplify this mechanism for the special case in which $k=1$ and each agent cannot alter the length of its job. Returning to the general setting, we show in Section 5 that, for any $k>1$, this competitive ratio is a lower bound for deterministic mechanisms that do not pay agents. All formal proofs are delayed to the appendix. Finally, in Section 6, we discuss related work other than the directly relevant [3] and [14], before concluding with Section 7.

2 Non-Strategic Setting

In this section, we formally define the original, non-strategic setting, and recap previous results.

2.1 Formulation

There exists a single processor on which jobs can execute, and a set $N=\{1,\ldots,n\}$ of jobs, although this number is not known beforehand. Each job i is characterized by a tuple $\theta_i=(r_i,d_i,l_i,v_i)$, which denotes the release time, deadline, length of processing time required, and value, respectively. The space Θ_i of possible tuples is the same for each job and consists of all θ_i such that $r_i,d_i,l_i,v_i \in \Re_+$ (thus, the model of time is continuous). Each job is released at time r_i, at which point its three other characteristics are known. Nothing is known about the job before its arrival. Each deadline is firm (or, hard), which means that no value is obtained for a job that is completed after its deadline. Preemption of jobs is allowed, and it takes no time to switch between jobs. Thus, job i is completed if and only if the total time it executes on the processor before d_i is at least l_i.

Define the *value density* $\rho_i = \frac{v_i}{l_i}$ of job i to be the ratio of its value to its length. For an input $\theta = (\theta_1, \ldots, \theta_n)$, denote the maximum and minimum value densities as $\rho_{min} = \min_i \rho_i$ and $\rho_{max} = \max_i \rho_i$. The *importance ratio* is then defined to be $\frac{\rho_{max}}{\rho_{min}}$, the maximal ratio of value densities between two jobs. The algorithm is assumed to always know an upper bound k on the importance ratio. For simplicity, we normalize the range of possible value densities so that $\rho_{min} = 1$.

An online algorithm is a function $f : \Theta_1 \times \ldots \times \Theta_n \to X$ that maps the vector of tuples (for any number n) to an alternative x. In this setting, an alternative $x \in X$ is simply a schedule of jobs on the processor, recorded by the function $\mathcal{S} : \Re_+ \to \{0, 1, \ldots, n\}$, which maps each point in time to the active job, or to 0 if the processor is idle. We will use $\mathcal{S}(\theta, t)$ as shorthand for the $\mathcal{S}(t)$ of $f(\theta)$, and it denotes the active job at time t when the input is θ.

To denote the total elapsed time that a job has spent on the processor at time t when the input is θ, we will use the function $e_i(\theta, t) = \int_0^t \mu(\mathcal{S}(\theta, x) = i)dx$, where $\mu(\cdot)$ is an indicator function that returns 1 if the argument is true, and zero otherwise. A job's *laxity* at time t is defined to be $\big(d_i - t - l_i + e_i(\theta, t)\big)$, the amount of time that it can remain inactive and still be completed by its deadline. A job is *abandoned* if it cannot be completed by its deadline (formally, if $d_i - t + e_i(\theta, t) < l_i$).

Since a job cannot be executed before its release time, the space of possible schedules is restricted in that $\mathcal{S}(\theta, t) = i$ implies $r_i \leq t$. Also, because the online algorithm must produce the schedule over time, without knowledge of future inputs, it must make the same decision at time t for inputs that are indistinguishable at this time. Formally, let $\theta(t)$ denote the subset of the tuples in θ that satisfy $r_i \leq t$. The constraint is then that $\theta(t) = \theta'(t)$ implies $\mathcal{S}(\theta, t) = \mathcal{S}(\theta', t)$.

The objective function is the sum of the values of the jobs that are completed by their respective deadlines: $W(f(\theta), \theta) = \sum_i \big(v_i \cdot \mu(e_i(\theta, d_i) \geq l_i)\big)$. Let $W^*(\theta) = \max_{x \in X} W(x, \theta)$ denote the maximum possible total value for the profile θ.

In competitive analysis, an online algorithm is evaluated by comparing it against an optimal offline algorithm. Because the offline algorithm knows the entire input θ at time 0 (but still cannot start each job i until time r_i), it always achieves $W^*(\theta)$. An online algorithm $f(\cdot)$ is (strictly) *c-competitive* if there does not exist an input θ such that $c \cdot W(f(\theta), \theta) < W^*(\theta)$. An algorithm that is c-competitive is also said to achieve a *competitive ratio* of c.

Finally, it is assumed that there does not exist an overload period of infinite duration. A period of time $[t^s, t^f]$ is overloaded if the sum of the lengths of the jobs whose release time and deadline both fall within the time period exceeds the duration of the interval (formally, if $t^f - t^s \leq \sum_{i|(t^s \leq r_i, d_i \leq t^f)} l_i$). Without such an assumption, it is not possible to achieve a finite competitive ratio [14].

2.2 Previous Results

In the non-strategic setting, [3] presents a 4-competitive algorithm called TD_1 (version 2) for the case of $k = 1$, while [14] presents a $(1 + \sqrt{k})^2$-competitive algorithm called D^{over} for the general case of $k \geq 1$. Matching lower bounds for deterministic algorithms for both of these cases were shown in [3]. In this section we provide a high-level description of TD_1 (version 2) using an example.

TD_1 (version 2) divides the schedule into intervals, each of which begins when the processor transitions from idle to busy (call this time t^b), and ends with the completion of a job. The first active job of an interval may have laxity; however, for the remainder of the interval, preemption of the active job is only considered when some other job has zero laxity. For example, when the input is the set of jobs listed in Table 1, the first interval is the complete execution of job 1 over the range $[0.0, 0.9]$. No preemption is considered during this interval, because job 2 does not have zero laxity before time 1.5. Then, a new interval starts at $t^b = 0.9$ when job 2 becomes active. Before job 2 can complete, preemption is considered at time 4.8, when job 3 is released with zero laxity.

In order to decide whether to preempt the active job, TD_1 (version 2) uses two more variables: t^e and *p_loss*. The former records the latest deadline of a job that would be abandoned if the active job executes to completion (or, if no such job exists, the time that the active job will finish if it is not preempted). In this case, $t^e = 17.0$. The value $t^e - t^b$ represents an upper bound on the amount of possible execution time "lost" to the optimal offline algorithm due to the completion of the active job. The other variable, p_loss, is equal to the length of the first active job of the current interval. Because in general this job could have laxity, the offline algorithm may be able to complete it outside of the range $[t^b, t^e]$.[1] If the algorithm completes the active job and this job's length is at least $\frac{t^e - t^b + p\_loss}{4}$, then the algorithm is guaranteed to be 4-competitive for this interval (note that $k = 1$ implies that all jobs have the same value density and thus that lengths can used to compute the competitive ratio). Because this is not case at time 4.8 (since $\frac{t^e - t^b + p\_loss}{4} = \frac{17.0 - 0.9 + 4.0}{4} > 4.0 = l_2$), the algorithm preempts job 2 for job 3, which then executes to completion.

Job	r_i	d_i	l_i	v_i
1	0.0	0.9	0.9	0.9
2	0.5	5.5	4.0	4.0
3	4.8	17.0	12.2	12.2

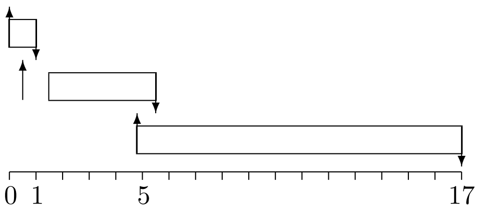


Table 1: Input used to recap TD_1 (version 2) [3]. The up and down arrows represent r_i and d_i, respectively, while the length of the box equals l_i.

3 Mechanism Design Setting

However, false information about job 2 would cause TD_1 (version 2) to complete this job. For example, if job 2's deadline were declared as $\hat{d}_2 = 4.7$, then it would have zero laxity at time 0.7. At this time, the algorithm would preempt job 1 for job 2, because $\frac{t^e - t^b + p\_loss}{4} = \frac{4.7 - 0.0 + 1.0}{4} > 0.9 = l_1$. Job 2 would then complete before the arrival of job 3.[2]

In order to address incentive issues such as this one, we need to formalize the setting as a mechanism design problem. In this section we first present the mechanism design formulation, and then define our goals for the mechanism.

3.1 Formulation

There exists a center, who controls the processor, and a set $N = \{1, \ldots, n\}$ of agents, where the value of n is unknown by the center beforehand. Each job i is owned by a separate agent i. The characteristics of the job define the agent's type $\theta_i \in \Theta_i$. At time r_i, agent i privately observes its type θ_i, and has no information about job i before r_i. Thus, jobs are still released over time, but now each job is released only to the owning agent.

Agents interact with the center through a direct mechanism $\Gamma = (\Theta_1, \ldots, \Theta_n, g(\cdot))$, in which each agent declares a job, denoted by $\hat{\theta}_i = (\hat{r}_i, \hat{d}_i, \hat{l}_i, \hat{v}_i)$, and the function $g : \Theta_1 \times \ldots \times \Theta_n \to O$ maps the declared types to an outcome $o \in O$. An outcome $o = (f(\cdot), p_1, \ldots, p_n)$ consists of the online algorithm $f(\cdot)$ that produces the schedule, and a payment from each agent to the mechanism.

[1]While it would be easy to alter the algorithm to recognize that this is not possible for the jobs in Table 1, our example does not depend on the use of p_loss.

[2]While we will not describe the significantly more complex D^{over}, we note that it is similar in its use of intervals and its preference for the active job. Also, we note that the lower bound we will show in Section 5 implies that false information can also benefit a job in D^{over}.

In a standard mechanism design setting, the outcome is enforced at the end of the mechanism. However, since the end is not well-defined in this online setting, we choose to model, for each agent i, the return of a completed job and the collection of a payment as occurring at $\hat{d}_i$. (which, according to agent i's declaration, is latest relevant point of time for that agent). Thus, even if job i is completed before $\hat{d}_i$, the center does not return the job to agent i until that time. This modelling decision could instead be viewed as a decision by the mechanism designer from a larger space of possible mechanisms. Indeed, as we will discuss later, this decision of when to return a completed job is crucial to our mechanism.

The utility function each agent aims to maximize, $u_i(g(\hat{\theta}), \theta_i) = v_i \cdot \mu(e_i(\hat{\theta}, d_i) \geq l_i) \cdot \mu(\hat{d}_i \leq d_i) - p_i(\hat{\theta})$, is a linear function of its value for its job (if completed and returned by its true deadline) and the payment it makes to the center.

Agent declarations are restricted in that an agent cannot declare a length shorter than the true length, since the center would be able to detect such a lie if the job were completed. On the other hand, in the general formulation we will allow agents to declare longer lengths, since in some settings it may be possible add unnecessary work to a job. However, we will also consider a restricted formulation in which this type of lie is not possible. The declared release time $\hat{r}_i$ is the time that the agent chooses to submit job i to the center, and it cannot precede the time r_i at which the job is revealed to the agent. The agent can declare an arbitrary deadline or value. To summarize, agent i can declare any type $\hat{\theta}_i = (\hat{r}_i, \hat{d}_i, \hat{l}_i, \hat{v}_i)$ such that $\hat{l}_i \geq l_i$ and $\hat{r}_i \geq r_i$.

While in the non-strategic setting it was sufficient for the algorithm to know the upper bound k on the ratio $\frac{\rho_{max}}{\rho_{min}}$, in the mechanism design setting we will strengthen this assumption so that the mechanism also knows ρ_{min} (or, equivalently, the range $[\rho_{min}, \rho_{max}]$ of possible value densities).[3] While we feel that it is unlikely that a center would know k without knowing this range, we later present a mechanism that does not depend on this extra knowledge in a restricted setting.

The restriction on the schedule is now that $\mathcal{S}(\hat{\theta}, t) = i$ implies $\hat{r}_i \leq t$, to capture the fact that a job cannot be scheduled on the processor before it is declared to the mechanism. As before, preemption of jobs is allowed, and job switching takes no time.

The constraints due to the online mechanism's lack of knowledge of the future are that $\hat{\theta}(t) = \hat{\theta}'(t)$ implies $\mathcal{S}(\hat{\theta}, t) = \mathcal{S}(\hat{\theta}', t)$, and $\hat{\theta}(\hat{d}_i) = \hat{\theta}'(\hat{d}_i)$ implies $p_i(\hat{\theta}) = p_i(\hat{\theta}')$ for each agent i. The setting can then be summarized as follows.

Setting 1 Overview

```
for all $t$ do
  The center instantiates $\mathcal{S}(\hat{\theta}, t) \leftarrow i$, for some $i$ s.t. $\hat{r}_i \leq t$
  if $\exists i, (r_i = t)$ then
    $\theta_i$ is revealed to agent $i$
  if $\exists i, (t \geq r_i)$ and agent $i$ has not declared a job then
    Agent $i$ can declare any job $\hat{\theta}_i$, s.t. $\hat{r}_i = t$ and $\hat{l}_i \geq l_i$
  if $\exists i, (\hat{d}_i = t) \wedge (e_i(\hat{\theta}, t) \geq l_i)$ then
    Completed job $i$ is returned to agent $i$
  if $\exists i, (\hat{d}_i = t)$ then
    Center sets and collects payment $p_i(\hat{\theta})$ from agent $i$
```

3.2 Mechanism Goals

Our aim as mechanism designer is to maximize the value of completed jobs, subject to the constraints of (dominant strategy) incentive compatibility and individual rationality, for which we use

[3]Note that we could then force agent declarations to satisfy $\rho_{min} \leq \frac{\hat{v}_i}{\hat{l}_i} \leq \rho_{max}$. However, this restriction would not decrease the lower bound on the competitive ratio.

the standard definitions.[4]

Definition 1 *A direct mechanism satisfies incentive compatibility (IC) if:*
$\forall i, \theta_i, \theta'_i, \hat{\theta}_{-i} : u_i(g(\theta_i, \hat{\theta}_{-i}), \theta_i) \geq u_i(g(\theta'_i, \hat{\theta}_{-i}), \theta_i)$

Definition 2 *A direct mechanism satisfies individual rationality (IR) if*
$\forall i, \theta_i, \hat{\theta}_{-i},\ u_i(g(\theta_i, \hat{\theta}_{-i}), \theta_i) \geq 0$

The social welfare function that we aim to maximize is the same as the objective function of the non-strategic setting: $W(f(\theta), \theta) = \sum_i \big(v_i \cdot \mu(e_i(\theta, d_i) \geq l_i)\big)$. As in the non-strategic setting, we will evaluate an online mechanism using competitive analysis to compare it against an optimal offline mechanism (which we will denote by $\Gamma_{offline}$). An offline mechanism knows all of the types at time 0, and thus can always achieve $W^*(\theta)$.[5]

Definition 3 *An online mechanism Γ is (strictly) c-competitive if it satisfies IC, and if there does not exist a profile of agent types θ such that $c \cdot W(f(\theta), \theta) < W^*(\theta)$.*

Note that, to guarantee that the competitive ratio has been achieved, the online mechanism must satisfy IC, in order to ensure that the declared types used by its algorithm are indeed the true types.

4 Results

In this section, we first present our main positive result: a $\big((1+\sqrt{k})^2+1\big)$-competitive mechanism ($\Gamma_1$). After providing some intuition as to why Γ_1 satisfies individual rationality and incentive compatibility, we formally prove first these two properties and then the competitive ratio. We then consider a special case in which $k = 1$ and agents cannot lie about the length of their job, which allows us to alter this mechanism so that it no longer requires either knowledge of ρ_{min} or the collection of payments from agents.

4.1 General Setting

For the full setting described above, we present Γ_1, which is formally defined below. Unlike TD_1 (version 2) and D^{over}, Γ_1 gives no preference to the active job. Instead, it always executes the available job with the highest *priority*: $(\hat{v}_i + \sqrt{k} \cdot e_i(\hat{\theta}, t) \cdot \rho_{min})$. Each agent whose job is completed is then charged the lowest value that it could have declared such that its job still would have been completed, holding constant the rest of its declaration.

We now state our theoretical results for this mechanism (with proofs found in the appendex), and provide intuition as to why Γ_1 addresses each of the incentive issues.

Theorem 1 *Mechanism Γ_1 satisfies IR.*

Theorem 2 *Mechanism Γ_1 satisfies IC.*

[4]A possible argument against the need for incentive compatibility in this setting is that an agent's lie may actually improve the schedule. In fact, this was the case in the example we showed for the false declaration $\hat{d}_2 = 4.7$. However, if an agent lies due to incorrect beliefs over the future input, then the lie could instead make the schedule the worse (for example, if job 3 were never released, then job 1 would have been unnecessarily abandoned). Furthermore, if we do not know the beliefs of the agents, and thus cannot predict how they will lie, then we can no longer provide a competitive guarantee for our mechanism.

[5]Another possibility is to allow only the agents to know their types at time 0, and to force $\Gamma_{offline}$ to be incentive compatible so that agents will truthfully declare their types at time 0. However, this would not affect our results, since executing a Clarke mechanism [8] at time 0 satisfies IC and IR, and always maximizes social welfare.

Mechanism 1 Γ_1

```
Execute $\mathcal{S}(\hat{\theta}, \cdot)$ according to Algorithm 1
for all i do
  if $e_i(\hat{\theta}, \hat{d}_i) \geq \hat{l}_i$    {Agent i's job is completed}    then
    $p_i(\hat{\theta}) \leftarrow \arg\min_{v'_i \geq 0}(e_i(((\hat{r}_i, \hat{d}_i, \hat{l}_i, v'_i), \hat{\theta}_{-i}), \hat{d}_i) \geq \hat{l}_i)$
  else
    $p_i(\hat{\theta}) \leftarrow 0$
```

Algorithm 1

```
for all $t$ do
  $Avail \leftarrow \{i | (t \geq \hat{r}_i) \wedge (e_i(\hat{\theta}, t) < \hat{l}_i) \wedge (e_i(\hat{\theta}, t) + \hat{d}_i - t \geq \hat{l}_i)\}$
     {Set of all released, non-completed, non-abandoned jobs}
  if $Avail \neq \emptyset$ then
    $\mathcal{S}(\hat{\theta}, t) \leftarrow \arg\max_{i \in Avail}(\hat{v}_i + \sqrt{k} \cdot e_i(\hat{\theta}, t) \cdot \rho_{min})$
             {Break ties in favor of lower $\hat{r}_i$}
  else
    $\mathcal{S}(\hat{\theta}, t) \leftarrow 0$
```

By the use of a payment rule similar to that of a second-price auction, Γ_1 satisfies both IC with respect to values and IR. We now argue why it satisfies IC with respect to the other three characteristics. Declaring an "improved" job (i.e., declaring an earlier release time, a shorter length, or a later deadline) could possibly decrease the payment of an agent. However, the first two lies are not possible in our setting, while the third would cause the job, if it is completed, to be returned to the agent after the true deadline. This is the reason why it is important to always return a completed job at its declared deadline, instead of at the point at which it is completed.

It remains to argue why an agent does not have incentive to "worsen" its job. First, note that if the job is completed both for truthful and a worse, false declaration, then the payment of the agent cannot decrease, since the completion of a job is monotonic in each of $\hat{r}_i$, $\hat{d}_i$, and $\hat{l}_i$. Second, the only possible effects of an inflated length on the completion of a job are to delay it or cause it to be abandoned, and the only possible effects of an earlier declared deadline are to cause it to be abandoned or to cause it to be returned earlier (which has no effect on the agent's utility in our setting). On the other hand, it is less obvious why agents do not have incentive to declare a later release time. Consider a mechanism Γ'_1 that differs from Γ_1 in that it does not preempt the active job i unless there exists another job j such that $(\hat{v}_i + \sqrt{k} \cdot l_i(\hat{\theta}, t) \cdot \rho_{min}) < \hat{v}_j$. Note that as an active job approaches completion in Γ_1, its condition for preemption approaches that of Γ'_1.

However, the types in Table 2 for the case of $k = 1$ show why an agent may have incentive to delay the arrival of its job under Γ'_1. Job 1 becomes active at time 0, and job 2 is abandoned upon its release at time 6, because $10 + 10 = v_1 + l_1 > v_2 = 13$. Then, at time 8, job 1 is preempted by job 3, because $10 + 10 = v_1 + l_1 < v_3 = 22$. Job 3 then executes to completion, forcing job 1 to be abandoned. However, job 2 had more "weight" than job 1, and would have prevented job 3 from being executed if it had been the active job at time 8, since $13 + 13 = v_2 + l_2 > v_3 = 22$. Thus, if agent 1 had falsely declared $\hat{r}_1 = 20$, then job 3 would have been abandoned at time 8, and job 1 would have completed over the range $[20, 30]$.

Intuitively, Γ_1 avoids this problem because of two properties. First, when a job becomes active, it must have a greater priority than all other available jobs. Second, because a job's priority can only increase through the increase of the term $(\sqrt{k} \cdot e_i(\hat{\theta}, t) \cdot \rho_{min})$, the rate of increase of a job's priority is independent of its characteristics. These two properties together imply that, while a job is active, there cannot exist a time at which its priority is less than the priority that one of these other jobs would have achieved by executing on the processor instead.

Job	r_i	d_i	l_i	v_i
1	0	30	10	10
2	6	19	13	13
3	8	30	22	22

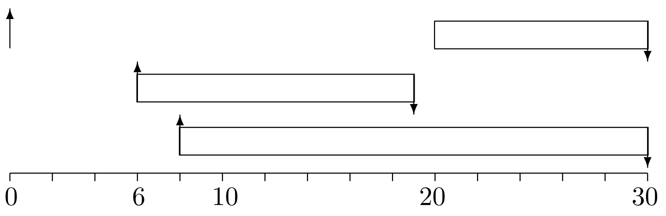


Table 2: Jobs used to show why a slightly altered version of Γ_1 would not be incentive compatible with respect to release times.

Using the fact that IC is satisfied, we can now prove that Γ_1 is $\big((1+\sqrt{k})^2+1\big)$-competitive by proving that the algorithm used by Γ_1 achieves this competitive ratio, assuming truthful inputs.

Theorem 3 *Mechanism* Γ_1 *is* $\big((1+\sqrt{k})^2+1\big)$*-competitive.*

4.2 Special Case: Unalterable length and k=1

While so far we have allowed each agent to lie about all four characteristics of its job, lying about the length of the job is not possible in some settings. For example, a user may not know how to alter a computational problem in a way that both lengthens the job and allows the solution of the original problem to be extracted from the solution to the altered problem. Another restriction that is natural in some settings is uniform value densities ($k = 1$), which was the case considered by [3]. If the setting satisfies these two conditions, then, by using Mechanism Γ_2 (formally described below), we can achieve a competitive ratio of 5 (which is the same competitive ratio as Γ_1 for the case of $k = 1$) without knowledge of ρ_{min} and without the use of payments. The latter property may be necessary in settings that are more local than grid computing (e.g., within a company) but in which the users are still self-interested.[6]

Mechanism 2 Γ_2

```
Execute $\mathcal{S}(\hat{\theta}, \cdot)$ according to Algorithm 2
for all i do
  $p_i(\hat{\theta}) \leftarrow 0$
```

Algorithm 2

```
for all $t$ do
  $Avail \leftarrow \{i | (t \geq \hat{r}_i) \wedge (e_i(\hat{\theta}, t) < l_i) \wedge (e_i(\hat{\theta}, t) + \hat{d}_i - t \geq l_i)\}$
  if $Avail \neq \emptyset$ then
    $\mathcal{S}(\hat{\theta}, t) \leftarrow \arg\max_{i \in Avail}(l_i + e_i(\hat{\theta}, t))$
              {Break ties in favor of lower $\hat{r}_i$}
  else
    $\mathcal{S}(\hat{\theta}, t) \leftarrow 0$
```

[6]While payments are not required in this setting, Γ_2 can be changed to collect a payments without affecting incentive compatibility by charging some fixed fraction of l_i for each job i that is completed.

Theorem 4 *When $k = 1$, and each agent i cannot falsely declare l_i, Mechanism Γ_2 satisfies IR and IC.*

Theorem 5 *When $k = 1$, and each agent i cannot falsely declare l_i, Mechanism Γ_2 is 5-competitive.*

Since this mechanism is essentially a simplification of Γ_1, we omit proofs of these theorems. Basically, the fact that $k = 1$ and $\hat{l}_i = l_i$ both hold allows Γ_2 to substitute the priority $(l_i + e_i(\hat{\theta}, t))$ for the priority used in Γ_1; and, since $\hat{v}_i$ is ignored, payments are no longer needed to ensure incentive compatibility.

5 Competitive Lower Bound

We now show that the competitive ratio of $(1 + \sqrt{k})^2 + 1$ achieved by Γ_1 is a lower bound for deterministic online mechanisms, under a pair of conditions. First, we appeal to third requirement on a mechanism, *non-negative payments* (NNP), which requires that the center never pays an agent (formally, $\forall i, \hat{\theta},\ p_i(\hat{\theta}_i) \geq 0$). While we did not require that our mechanisms satisfy this requirement, we note that both Γ_1 and Γ_2 satisfy it trivially, and that, in the proof of this theorem (found in the appendix), zero only serves as a baseline utility for an agent, and could be replaced by any non-positive function of $\hat{\theta}_{-i}$.

Second, we restrict consideration to settings in which $k > 1$. For the case of $k = 1$, we can achieve a competitive ratio of 4 by using TD_1 (version 2) [3], and charging $\hat{l}_i \cdot \rho_{min}$ to each agent i whose job is completed. Any agent who truthfully declares the length of his job is indifferent between whether his job is completed or not, because $k = 1$ implies that his payment upon completion will be equal to his value. If he inflates the length of his job, then he will have negative utility for its completion. Thus, an agent has no incentive to falsely declare his type. However, we chose not to use this mechanism for the restricted setting in the previous section, because agents never have incentive to even participate (or not to declare a type such that their job would never be completed).

Theorem 6 *There does not exist a deterministic online mechanism that satisfies IC, IR, and NNP, and that achieves a competitive ratio less than $(1 + \sqrt{k})^2 + 1$, for any $k > 1$.*

6 Related Work

In this section we describe related work other than the two papers ([3] and [14]) on which this work is based. Recent work related to this scheduling domain has focused on competitive analysis in which the online algorithm uses a faster processor than the offline algorithm (see, e.g., [12, 13]). Mechanism design was also applied to a scheduling problem in [16]. In their model, the center owns the jobs in an offline setting, and it is the agents who can execute them. The private information of an agent is the time it will require to execute each job. Several incentive compatible mechanisms are presented that are based on approximation algorithms for the computationally infeasible optimization problem.

Online execution presents a different type of algorithmic challenge, and several other papers study online algorithms or mechanisms in economic settings. For example, [4] considers an online market clearing setting, in which the auctioneer matches buy and sells bids (which are assumed to be exogenous) that arrive and expire over time. In [1], a general method is presented for converting an online algorithm into an online mechanism that is incentive compatible with respect to values. Truthful declaration of values is also considered in [2] and [15], which both consider multi-unit online auctions. The main difference between the two is that the former considers the case of a digital good, which thus has unlimited supply. It is pointed out in [15] that their results continue to hold when the setting is extended so that bidders can delay their arrival.

The only other work we are aware of that addresses the issue of incentive compatibility in a real-time system is [10], which considers several variants of a model in which the center allocates bandwidth to agents who declare both their value and their arrival time. A dominant strategy IC mechanism is presented for the variant in which every point in time is essentially independent, while a Bayes-Nash IC mechanism is presented for the variant in which the center's current decision affects the cost of future actions.

7 Discussion

In this paper, we considered an online scheduling domain for which algorithms with the best possible competitive ratio had been found, but for which new solutions were required when the setting is extended to include self-interested agents. We presented a mechanism that is incentive compatible with respect to release time, deadline, length and value, and that only increases the competitive ratio by one. We also showed how this mechanism could be simplified when $k = 1$ and each agent cannot lie about the length of its job. We then showed a matching lower bound, under a pair of conditions, on the competitive ratio that can be achieved by a deterministic mechanism.

It would be interesting to determine whether the lower bound can be strengthened by removing the restriction of non-negative payments. More generally, the use of randomized mechanisms in this setting provides an unexplored area for future work.

References

[1] B. Awerbuch, Y. Azar, and A. Meyerson, *Reducing truth-telling online mechanisms to online optimization*, Proceedings on the 35th Symposium on the Theory of Computing, 2003.

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

[3] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, and F. Wang, *On the competitiveness of on-line real-time task scheduling*, Journal of Real-Time Systems **4** (1992), no. 2, 125–144.

[4] A. Blum, T. Sandholm, and M. Zinkevich, *Online algorithms for market clearing*, Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, 2002.

[5] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*, Cambridge University Press, 1998.

[6] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, *Economic models for resource management and scheduling in grid computing*, The Journal of Concurrency and Computation: Practice and Experience **14** (2002), 1507–1542.

[7] N. Camiel, S. London, N. Nisan, and O. Regev, *The popcorn project: Distributed computation over the internet in java*, 6th International World Wide Web Conference, 1997.

[8] E. Clarke, *Multipart pricing of public goods*, Public Choice **18** (1971), 19–33.

[9] A. Fiat and G. Woeginger (editors), *Online algorithms: The state of the art*, Springer Verlag, 1998.

[10] E. Friedman and D. Parkes, *Pricing wifi at starbucks- issues in online mechanism design*, EC'03, 2003.

[11] R. L. Graham, *Bounds for certain multiprocessor anomalies*, Bell System Technical Journal **45** (1966), 1563–1581.

[12] B. Kalyanasundaram and K. Pruhs, *Speed is as powerful as clairvoyance*, Journal of the ACM **47** (2000), 617–643.

[13] C. Koo, T. Lam, T. Ngan, and K. To, *On-line scheduling with tight deadlines*, Theoretical Computer Science **295** (2003), 251–261.

[14] G. Koren and D. Shasha, *D-over: An optimal on-line scheduling algorithm for overloaded real-time systems*, SIAM Journal of Computing **24** (1995), no. 2, 318–339.

[15] R. Lavi and N. Nisan, *Competitive analysis of online auctions*, EC'00, 2000.

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

[17] K. Rosen, *Discrete mathematics and its applications*, 2nd ed., McGraw-Hill, Inc., 1995.

A Proofs

A.1 Proof of Theorem 1

Theorem 7 *Mechanism Γ_1 satisfies IR.*

Proof: For arbitrary $i, \theta_i, \hat{\theta}_{-i}$, if job i is not completed, then agent i pays nothing and thus has a utility of zero; that is, $p_i(\theta_i, \hat{\theta}_{-i}) = 0$ and $u_i(g(\theta_i, \hat{\theta}_{-i}), \theta_i) = 0$. On the other hand, if job i is completed, then its value must exceed agent i's payment. Formally, $u_i(g(\theta_i, \hat{\theta}_{-i}), \theta_i) = v_i - \arg\min_{v'_i \geq 0}(e_i(((r_i, d_i, l_i, v'_i), \hat{\theta}_{-i}), d_i) \geq l_i) \geq 0$ must hold, since $v'_i = v_i$ satisfies the condition. □

A.2 Proof of Theorem 2

To prove incentive compatibility, we need to show that for an arbitrary agent i with type θ_i, and an arbitrary profile $\hat{\theta}_{-i}$ of declarations of the other agents, agent i can never gain by making a false declaration $\hat{\theta}_i \neq \theta_i$, subject to the constraints that $\hat{r}_i \geq r_i$ and $\hat{l}_i \geq l_i$. We break this proof into lemmas.

We start by showing that, regardless of $\hat{v}_i$, if truthful declarations of r_i, d_i, and l_i do not cause job i to be completed, then "worse" declarations of these variables (that is, declarations that satisfy $\hat{r}_i \geq r_i$, $\hat{l}_i \geq l_i$ and $\hat{d}_i \leq d_i$) can never cause the job to be completed. We break this part of the proof into two lemmas, first showing that it holds for the release time, regardless of the declarations of the other variables, and then for length and deadline.

Lemma 8 *In mechanism Γ_1, the following condition holds for all $i, \theta_i, \hat{\theta}_{-i}$:*

$$\forall\ \hat{v}_i,\ \hat{l}_i \geq l_i,\ \hat{d}_i \leq d_i,\ \hat{r}_i \geq r_i, \quad \left[e_i\big(((\hat{r}_i, \hat{d}_i, \hat{l}_i, \hat{v}_i), \hat{\theta}_{-i}), \hat{d}_i\big) \geq \hat{l}_i\right] \implies \left[e_i\big(((r_i, \hat{d}_i, \hat{l}_i, \hat{v}_i), \hat{\theta}_{-i}), \hat{d}_i\big) \geq \hat{l}_i\right]$$

Proof: Assume by contradiction that this condition does not hold– that is, job i is not completed when r_i is truthfully declared, but is completed for some false declaration $\hat{r}_i \geq r_i$. We first analyze the case in which the release time is truthfully declared, and then we show that job i cannot be completed when agent i delays submitting it to the center.

Case I: Agent i declares $\hat{\theta}'_i = (r_i, \hat{d}_i, \hat{l}_i, \hat{v}_i)$.

First, define the following three points in the execution of job i.

- Let $t^s = \arg\min_t \big(\mathcal{S}((\hat{\theta}'_i, \hat{\theta}_{-i}), t) = i\big)$ be the time that job i first starts execution.

- Let $t^p = \arg\min_{t > t^s} \big(\mathcal{S}((\hat{\theta}'_i, \hat{\theta}_{-i}), t) \neq i\big)$ be the time that job i is first preempted.
- Let $t^a = \arg\min_t \big(e_i((\hat{\theta}'_i, \hat{\theta}_{-i}), t) + \hat{d}_i - t < \hat{l}_i\big)$ be the time that job i is abandoned.

If t^s and t^p are undefined because job i never becomes active, then let $t^s = t^p = t^a$.
Also, partition the jobs declared by other agents before t^a into the following three sets.

- Let $X = \{j | (\hat{r}_j < t^p) \wedge (j \neq i)\}$ consist of the jobs (other than i) that arrive before job i is first preempted.
- Let $Y = \{j | (t^p \leq \hat{r}_j \leq t^a) \wedge (\hat{v}_j > \hat{v}_i + \sqrt{k} \cdot e_i((\hat{\theta}'_i, \hat{\theta}_{-i}), \hat{r}_j)\}$ consist of the jobs that arrive in the range $[t^p, t^a]$ and that when they arrive have higher priority than job i (note that we are make use of the normalization that $\rho_{min} = 1$).
- Let $Z = \{j | (t^p \leq \hat{r}_j \leq t^a) \wedge (\hat{v}_j \leq \hat{v}_i + \sqrt{k} \cdot e_i((\hat{\theta}'_i, \hat{\theta}_{-i}), \hat{r}_j)\}$ consist of the jobs that arrive in the range $[t^p, t^a]$ and that when they arrive have lower priority than job i.

We now show that all active jobs during the range $(t^p, t^a]$ must be either i or in the set Y. Unless $t^p = t^a$ (in which case this property trivially holds), it must be the case that job i has a higher priority than an arbitrary job $x \in X$ at time t^p, since at the time just preceding t^p job x was available and job i was active. Formally, $\hat{v}_x + \sqrt{k} \cdot e_x((\hat{\theta}'_i, \hat{\theta}_{-i}), t^p) < \hat{v}_i + \sqrt{k} \cdot e_i((\hat{\theta}'_i, \hat{\theta}_{-i}), t^p)$ must hold.[7] We can then show that, over the range $[t^p, t^a]$, no job $x \in X$ runs on the processor. Assume by contradiction that this is not true. Let $t^f \in [t^p, t^a]$ be the earliest time in this range that some job $x \in X$ is active, which implies that $e_x((\hat{\theta}'_i, \hat{\theta}_{-i}), t^f) = e_x((\hat{\theta}'_i, \hat{\theta}_{-i}), t^p)$. We can then show that job i has a higher priority at time t^f as follows: $\hat{v}_x + \sqrt{k} \cdot e_x((\hat{\theta}'_i, \hat{\theta}_{-i}), t^f) = \hat{v}_x + \sqrt{k} \cdot e_x((\hat{\theta}'_i, \hat{\theta}_{-i}), t^p) < \hat{v}_i + \sqrt{k} \cdot e_i((\hat{\theta}'_i, \hat{\theta}_{-i}), t^p) \leq \hat{v}_i + \sqrt{k} \cdot e_i((\hat{\theta}'_i, \hat{\theta}_{-i}), t^f)$, contradicting the fact that job x is active at time t^f.

A similar argument applies to an arbitrary job $z \in Z$, starting at it release time $\hat{r}_z > t^p$, since by definition job i has a higher priority at that time. The only remaining jobs that can be active over the range $(t^p, t^a]$ are i and those in the set Y.

Case II: Agent i declares $\hat{\theta}_i = (\hat{r}_i, \hat{d}_i, \hat{l}_i, \hat{v}_i)$, where $\hat{r}_i > r_i$.

We now show that job i cannot be completed in this case, given that it was not completed in case I. First, we can restrict the range of $\hat{r}_i$ that we need to consider as follows. Declaring $\hat{r}_i \in (r_i, t^s]$ would not affect the schedule, since t^s would still be the first time that job i executes. Also, declaring $\hat{r}_i > t^a$ could not cause the job to be completed, since $d_i - t^a < \hat{l}_i$ holds, which implies that job i would be abandoned at its release. Thus, we can restrict consideration to $\hat{r}_i \in (t^s, t^a]$.

In order for declaring $\hat{\theta}_i$ to cause job i to be completed, a necessary condition is that the execution of some job $y^c \in Y$ must change during the range $(t^p, t^a]$, since the only jobs other than i that are active during that range are in Y. Let $t^c = \arg\min_{t \in (t^p, t^a]} [\exists y^c \in Y, (\mathcal{S}((\hat{\theta}'_i, \hat{\theta}_{-i}), t) = y^c) \wedge (\mathcal{S}((\hat{\theta}_i, \hat{\theta}_{-i}), t) \neq y^c)]$ be the first time that such a change occurs. We will now show that for any $\hat{r}_i \in (t^s, t^a]$, there cannot exist a job with higher priority than y^c at time t^c, contradicting $(\mathcal{S}((\hat{\theta}_i, \hat{\theta}_{-i}), t) \neq y^c)$.

First note that job i cannot have a higher priority, since there would have to exist a $t \in (t^p, t^c)$ such that $\exists y \in Y, (\mathcal{S}((\hat{\theta}'_i, \hat{\theta}_{-i}), t) = y) \wedge (\mathcal{S}((\hat{\theta}_i, \hat{\theta}_{-i}), t) = i)$, contradicting the definition of t^c.

Now consider an arbitrary $y \in Y$ such that $y \neq y^c$. In case I, we know that job y has lower priority than y^c at time t^c; that is, $\hat{v}_y + \sqrt{k} \cdot e_y((\hat{\theta}'_i, \hat{\theta}_{-i}), t^c) < \hat{v}_{y^c} + \sqrt{k} \cdot e_{y^c}((\hat{\theta}'_i, \hat{\theta}_{-i}), t^c)$. Thus, moving to case II, job y must replace some other job before t^c. Since $\hat{r}_y \geq t^p$, the condition is that there must exist some $t \in (t^p, t^c)$ such that $\exists w \in Y \cup \{i\}, (\mathcal{S}((\hat{\theta}'_i, \hat{\theta}_{-i}), t) = w) \wedge (\mathcal{S}((\hat{\theta}_i, \hat{\theta}_{-i}), t) = y)$.

[7]For simplicity, when we give the formal condition for a job x to have a higher priority than another job y, we will assume that job x's priority is strictly greater than job y's, because, in the case of a tie that favors x, future ties would also be broken in favor of job x.

Since $w \in Y$ would contradict the definition of t^c, we know that $w = i$. That is, the job that y replaces must be i. By definition of the set Y, we know that $\hat{v}_y > \hat{v}_i + \sqrt{k} \cdot e_i((\hat{\theta}'_i, \hat{\theta}_{-i}), \hat{r}_y)$. Thus, if $\hat{r}_y \leq t$, then job i could not have executed instead of y in case I. On the other hand, if $\hat{r}_y > t$, then job y obviously could not execute at time t, contradicting the existence of such a time t.

Now consider an arbitrary job $x \in X$. We know that in case I job i has a higher priority than job x at time t^s, or, formally, that $\hat{v}_x + \sqrt{k} \cdot e_x((\hat{\theta}'_i, \hat{\theta}_{-i}), t^s) < \hat{v}_i + \sqrt{k} \cdot e_i((\hat{\theta}'_i, \hat{\theta}_{-i}), t^s)$. We also know that $\hat{v}_i + \sqrt{k} \cdot e_i((\hat{\theta}'_i, \hat{\theta}_{-i}), t^c) < \hat{v}_{y^c} + \sqrt{k} \cdot e_{y^c}((\hat{\theta}'_i, \hat{\theta}_{-i}), t^c)$. Since delaying i's arrival will not affect the execution up to time t^s, and since job x cannot execute instead of a job $y \in Y$ at any time $t \in (t^p, t^c]$ by definition of t^c, the only way for job x's priority to increase before t^c as we move from case I to II is to replace job i over the range $(t^s, t^c]$. Thus, an upper bound on job x's priority when agent i declares $\hat{\theta}_i$ is: $\hat{v}_x + \sqrt{k} \cdot \big[e_x((\hat{\theta}'_i, \hat{\theta}_{-i}), t^s) + e_i((\hat{\theta}'_i, \hat{\theta}_{-i}), t^c) - e_i((\hat{\theta}'_i, \hat{\theta}_{-i}), t^s)\big] < \hat{v}_i + \sqrt{k} \cdot \big[e_i((\hat{\theta}'_i, \hat{\theta}_{-i}), t^s) + e_i((\hat{\theta}'_i, \hat{\theta}_{-i}), t^c) - e_i((\hat{\theta}'_i, \hat{\theta}_{-i}), t^s)\big] = \hat{v}_i + \sqrt{k} \cdot e_i((\hat{\theta}'_i, \hat{\theta}_{-i}), t^c) < \hat{v}_{y^c} + \sqrt{k} \cdot e_{y^c}((\hat{\theta}'_i, \hat{\theta}_{-i}), t^c)$.

Thus, even at this upper bound, job y^c would execute instead of job x at time t^c. A similar argument applies to an arbitrary job $z \in Z$, starting at it release time $\hat{r}_z$. Since the sets $\{i\}, X, Y, Z$ partition the set of jobs released before t^a, we have shown that no job could execute instead of job y^c, contradicting the existence of t^c, and completing the proof. □

Lemma 9 *In mechanism Γ_1, the following condition holds for all $i, \theta_i, \hat{\theta}_{-i}$:*

$$\forall\ \hat{v}_i,\ \hat{l}_i \geq l_i,\ \hat{d}_i \leq d_i, \quad \big[e_i\big(((r_i, \hat{d}_i, \hat{l}_i, \hat{v}_i), \hat{\theta}_{-i}), \hat{d}_i\big) \geq \hat{l}_i\big] \implies \big[e_i\big(((r_i, d_i, l_i, \hat{v}_i), \hat{\theta}_{-i}), d_i\big) \geq l_i\big]$$

Proof: Assume by contradiction there exists some instantiation of the above variables such that job i is not completed when l_i and d_i are truthfully declared, but is completed for some pair of false declarations $\hat{l}_i \geq l_i$ and $\hat{d}_i \leq d_i$.

Note that the only effect that $\hat{d}_i$ and $\hat{l}_i$ have on the execution of the algorithm is on whether or not $i \in \mathit{Avail}$. Specifically, they affect the two conditions: $(e_i(\hat{\theta}, t) < \hat{l}_i)$ and $(e_i(\hat{\theta}, t) + \hat{d}_i - t \geq \hat{l}_i)$. Because job i is completed when $\hat{l}_i$ and $\hat{d}_i$ are declared, the former condition (for completion) must become false before the latter. Since truthfully declaring $l_i \leq \hat{l}_i$ and $d_i \geq \hat{d}_i$ will only make the former condition become false earlier and the latter condition become false later, the execution of the algorithm will not be affected when moving to truthful declarations, and job i will be completed, a contradiction. □

We now use these two lemmas to show that the payment for a completed job can only increase by falsely declaring "worse" $\hat{l}_i$, $\hat{d}_i$, and $\hat{r}_i$.

Lemma 10 *In mechanism Γ_1, the following condition holds for all $i, \theta_i, \hat{\theta}_{-i}$:*

$$\forall\ \hat{l}_i \geq l_i,\ \hat{d}_i \leq d_i,\ \hat{r}_i \geq r_i, \quad \arg\min_{v'_i \geq 0} \big[e_i\big(((\hat{r}_i, \hat{d}_i, \hat{l}_i, v'_i), \hat{\theta}_{-i}), \hat{d}_i\big) \geq \hat{l}_i\big] \geq \arg\min_{v'_i \geq 0} \big[e_i\big(((r_i, d_i, l_i, v'_i), \hat{\theta}_{-i}), d_i\big) \geq l_i\big]$$

Proof: Assume by contradiction that this condition does not hold. This implies that there exists some value v'_i such that the condition $(e_i(((\hat{r}_i, \hat{d}_i, \hat{l}_i, v'_i), \hat{\theta}_{-i}), \hat{d}_i) \geq \hat{l}_i)$ holds, but $(e_i(((r_i, d_i, l_i, v'_i), \hat{\theta}_{-i}), d_i) \geq l_i)$ does not. Applying Lemmas 8 and 9: $(e_i(((\hat{r}_i, \hat{d}_i, \hat{l}_i, v'_i), \hat{\theta}_{-i}), \hat{d}_i) \geq \hat{l}_i) \implies (e_i(((r_i, \hat{d}_i, \hat{l}_i, v'_i), \hat{\theta}_{-i}), \hat{d}_i) \geq \hat{l}_i) \implies$
$(e_i(((r_i, d_i, l_i, v'_i), \hat{\theta}_{-i}), d_i) \geq l_i)$, a contradiction. □

Finally, the following lemma tells us that the completion of a job is monotonic in its declared value.

Lemma 11 *In mechanism Γ_1, the following condition holds for all $i, \hat{\theta}_i, \hat{\theta}_{-i}$:*

$$\forall\ \hat{l}_i \geq l_i,\ \hat{d}_i \leq d_i,\ \hat{r}_i \geq r_i,\ \hat{v}'_i \geq \hat{v}_i, \quad \big[e_i\big(((\hat{r}_i, \hat{d}_i, \hat{l}_i, \hat{v}_i), \hat{\theta}_{-i}), \hat{d}_i\big) \geq \hat{l}_i\big] \implies \big[e_i\big(((\hat{r}_i, \hat{d}_i, \hat{l}_i, \hat{v}'_i), \hat{\theta}_{-i}), \hat{d}_i\big) \geq \hat{l}_i\big]$$

The proof, by contradiction, of this lemma is omitted because it is essentially identical to that of Lemma 8 for $\hat{r}_i$. In case I, agent i declares $(\hat{r}_i, \hat{d}_i, \hat{l}_i, \hat{v}'_i)$ and the job is not completed, while in case II he declares $(\hat{r}_i, \hat{d}_i, \hat{l}_i, \hat{v}_i)$ and the job is completed. The analysis of the two cases then proceeds as before– the execution will not change up to time t^s because the initial priority of job i decreases as we move from case I to II; and, as a result, there cannot be a change in the execution of a job other than i over the range $(t^p, t^a]$.

We can now combine the lemmas to show that no profitable deviation is possible, proving Theorem 2.

Theorem 12 *Mechanism Γ_1 satisfies IC.*

Proof: For an arbitrary agent i, we know that $\hat{r}_i \geq r_i$ and $\hat{l}_i \geq l_i$ hold by assumption. We also know that agent i has no incentive to declare $\hat{d}_i > d_i$, because job i would never be returned before its true deadline. Then, because the payment function is non-negative, agent i's utility could not exceed zero. By IR, this is the minimum utility it would achieve if it truthfully declared θ_i. Thus, we can restrict consideration to $\hat{\theta}_i$ that satisfy $\hat{r}_i \geq r_i$, $\hat{l}_i \geq l_i$, and $\hat{d}_i \leq d_i$. Again using IR, we can further restrict consideration to $\hat{\theta}_i$ that cause job i to be completed, since any other $\hat{\theta}_i$ yields a utility of zero.

If truthful declaration of θ_i causes job i to be completed, then by Lemma 10 any such false declaration $\hat{\theta}_i$ could not decrease the payment of agent i. On the other hand, if truthful declaration does not cause job i to be completed, then declaring such a $\hat{\theta}_i$ will cause agent i to have negative utility, since, by Lemmas 11 and 10, it must be the case that: $v_i < \arg\min_{v'_i \geq 0} \left[e_i(((r_i, d_i, l_i, v'_i), \hat{\theta}_{-i}), \hat{d}_i) \geq l_i\right] \leq \arg\min_{v'_i \geq 0} \left[e_i(((\hat{r}_i, \hat{d}_i, \hat{l}_i, v'_i), \hat{\theta}_{-i}), \hat{d}_i) \geq \hat{l}_i\right]$. □

A.3 Proof of Theorem 3

The proof of the competitive ratio, which makes use of techniques adapted from those used in [14], is also broken into lemmas. Having shown IC, we can assume truthful declaration ($\hat{\theta} = \theta$), and it remains to bound the loss of social welfare against $\Gamma_{offline}$.

Denote by $(1, 2, \ldots, F)$ the sequence of jobs completed by Γ_1. Divide time into intervals $I_f = (t_f^{open}, t_f^{close}]$, one for each job f in this sequence. Set t_f^{close} to be the time at which job f is completed, and set $t_f^{open} = t_{f-1}^{close}$ for $f \geq 2$, and $t_1^{open} = 0$ for $f = 1$. Also, let t_f^{begin} be the first time that the processor is not idle in interval I_f.

Lemma 13 *For any interval I_f, the following inequality holds: $t_f^{close} - t_f^{begin} \leq (1 + \frac{1}{\sqrt{k}}) \cdot v_f$*

Proof: Interval I_f begins with a (possibly zero length) period of time in which the processor is idle because there is no available job. Then, it continuously executes a sequence of jobs $(1, 2, \ldots, c)$, where each job i in this sequence is preempted by job $i + 1$, except for job c, which is completed (thus, job c in this sequence is the same as job f is the global sequence of completed jobs). Let t_i^s be the time that job i begins execution. Note that $t_1^s = t_f^{begin}$.

Over the range $[t_f^{begin}, t_f^{close}]$, the priority $(v_i + \sqrt{k} \cdot e_i(\theta, t))$ of the active job is monotonically increasing with time, because this function linearly increases while a job is active, and can only increase at a point in time when preemption occurs. Thus, each job $i > 1$ in this sequence begins execution at its release time (that is, $t_i^s = r_i$), because its priority does not increase while it is not active.

We now show that the value of the completed job c exceeds the product of $\sqrt{k}$ and the time spent in the interval on jobs 1 through $c - 1$, or, more formally, that the following condition holds: $v_c \geq \sqrt{k} \sum_{h=1}^{c-1} (e_h(\theta, t_{h+1}^s) - e_h(\theta, t_h^s))$. To show this, we will prove by induction that the stronger condition $v_i \geq \sqrt{k} \sum_{h=1}^{i-1} e_h(\theta, t_{h+1}^s)$ holds for all jobs i in the sequence.

Base Case: For $i = 1$, $v_1 \geq \sqrt{k} \sum_{h=1}^{0} e_h(\theta, t^s_{h+1}) = 0$, since the sum is over zero elements.

Inductive Step: For an arbitrary $1 \leq i < c$, we assume that $v_i \geq \sqrt{k} \sum_{h=1}^{i-1} e_h(\theta, t^s_{h+1})$ holds. At time t^s_{i+1}, we know that $v_{i+1} \geq v_i + \sqrt{k} \cdot e_i(\theta, t^s_{i+1})$ holds, because $t^s_{i+1} = r_{i+1}$. These two inequalities together imply that $v_{i+1} \geq \sqrt{k} \sum_{h=1}^{i} e_h(\theta, t^s_{h+1})$, completing the inductive step.

We also know that $t^{close}_f - t^s_c \leq l_c \leq v_c$ must hold, by the simplifying normalization of $\rho_{min} = 1$ and the fact that job c's execution time cannot exceed its length. We can thus bound the total execution time of I_f by: $t^{close}_f - t^{begin}_f = (t^{close}_f - t^s_c) + \sum_{h=1}^{c-1} (e_h(\theta, t^s_{h+1}) - e_h(\theta, t^s_h)) \leq (1 + \frac{1}{\sqrt{k}}) v_f$.
$\square$

We now consider the possible execution of uncompleted jobs by $\Gamma_{offline}$. Associate each job i that is not completed by Γ_1 with the interval during which it was abandoned. All jobs are now associated with an interval, since there are no gaps between the intervals, and since no job i can be abandoned after the close of the last interval at t^{close}_F. Because the processor is idle after t^{close}_F, any such job i would become active at some time $t \geq t^{close}_F$, which would lead to the completion of some job, creating a new interval and contradicting the fact that I_F is the last one.

The following lemma is equivalent to Lemma 5.6 of [14], but the proof is different for our mechanism.

Lemma 14 *For any interval I_f and any job i abandoned in I_f, the following inequality holds: $v_i \leq (1 + \sqrt{k}) v_f$.*

Proof: Assume by contradiction that there exists a job i abandoned in I_f such that $v_i > (1+\sqrt{k}) v_f$. At t^{close}_f, the priority of job f is $v_f + \sqrt{k} \cdot l_f < (1 + \sqrt{k}) v_f$. Because the priority of the active job monotonically increases over the range $[t^{begin}_f, t^{close}_f]$, job i would have a higher priority than the active job (and thus begin execution) at some time $t \in [t^{begin}_f, t^{close}_f]$. Again applying monotonicity, this would imply that the priority of the active job at t^{close}_f exceeds $(1 + \sqrt{k}) v_f$, contradicting the fact that it is $(1 + \sqrt{k}) v_f$. $\square$

As in [14], for each interval I_f, we give $\Gamma_{offline}$ the following "gift": k times the amount of time in the range $[t^{begin}_f, t^{close}_f]$ that it does not schedule a job. Additionally, we "give" the adversary v_f, since the adversary may be able to complete this job at some future time, due to the fact that Γ_1 ignores deadlines. The following lemma is Lemma 5.10 in [14], and its proof now applies directly.

Lemma 15 *[14] With the above gifts the total net gain obtained by the clairvoyant algorithm from scheduling the jobs abandoned during I_f is not greater than $(1 + \sqrt{k}) \cdot v_f$.*

The intuition behind this lemma is that the best that the adversary can do is to take almost all of the "gift" of $k \cdot (t^{close}_f - t^{begin}_f)$ (intuitively, this is equivalent to executing jobs with the maximum possible value density over the time that Γ_1 is active), and then begin execution of a job abandoned by Γ_1 right before t^{close}_f. By Lemma 14, the value of this job is bounded by $(1 + \sqrt{k}) \cdot v_f$. We can now combine the results of these lemmas to prove Theorem 3.

Theorem 16 *Mechanism Γ_1 is $((1 + \sqrt{k})^2 + 1)$-competitive.*

Proof: Using the fact that the way in which jobs are associated with the intervals partitions the entire set of jobs, we can show the competitive ratio by showing that Γ_1 is $\big((1 + \sqrt{k})^2 + 1\big)$-competitive for each interval in the sequence $(1, \ldots, F)$. Over an arbitrary interval I_f, the offline algorithm can achieve at most $(t^{close}_f - t^{begin}_f) \cdot k + v_f + (1 + \sqrt{k}) v_f$, from the two gifts and the net gain bounded by Lemma 15. Applying Lemma 13, this quantity is then bounded from above by $(1 + \frac{1}{\sqrt{k}}) \cdot v_f \cdot k + v_f + (1 + \sqrt{k}) v_f = ((1 + \sqrt{k})^2 + 1) \cdot v_f$. Since Γ_1 achieves v_f, the competitive ratio holds. $\square$

A.4 Proof of Theorem 6

The proof the lower bound uses an adversary argument similar to that used in [3] to show a lower bound of $(1+\sqrt{k})^2$ in the non-strategic setting, with the main novelty lying in the two perturbations of the job sequence and the related incentive compatibility arguments. We first prove a lemma relating to the recurrence used for this argument.

Lemma 1 *For any $k \geq 1$, for the recurrence defined by:*

$$\begin{aligned} l_{j+1} &= \lambda \cdot l_j - k \cdot \sum_{h=1}^{j} l_h \qquad (1) \\ l_1 &= 1 \end{aligned}$$

where $(1+\sqrt{k})^2 - 1 < \lambda < (1+\sqrt{k})^2$, there exists an integer $m \geq 1$ such that:

$$\frac{l_m + k \cdot \sum_{h=1}^{m-1} l_h}{l_m} > \lambda \qquad (2)$$

Proof: This proof is a generalization of the one shown in [3] for the case of $k = 1$. We first show that the existence of such a number m is equivalent to the existence of an $m \geq 1$ such that $l_m < l_{m-1}$. Rearranging Equation 2, and using Equation 1 to substitute in for l_m yields:

$$\begin{aligned} l_m + k \cdot \sum_{h=1}^{m-1} l_h &> \lambda \cdot l_m \\ \lambda \cdot l_{m-1} - k \cdot \sum_{h=1}^{m-1} l_h + k \cdot \sum_{h=1}^{m-1} l_h &> \lambda \cdot l_m \\ l_{m-1} &> l_m \end{aligned}$$

We now show that there exists an $m \geq 1$ that satisfies $l_m < l_{m-1}$. Substituting j in for $j+1$ in Equation 1 yields:

$$l_j = \lambda \cdot l_{j-1} - k \cdot \sum_{h=1}^{j-1} l_h \qquad (3)$$

Subtracting Equation 3 from Equation 1 produces:

$$\begin{aligned} l_{j+1} - l_j &= \lambda \cdot l_j - \lambda \cdot l_{j-1} - k \cdot l_j \\ l_{j+1} &= (\lambda + 1 - k) \cdot l_j - \lambda \cdot l_{j-1} \end{aligned}$$

Thus, we can re-write the recurrence as:

$$\begin{aligned} l_{j+2} &= (\lambda + 1 - k) \cdot l_{j+1} - \lambda \cdot l_j \\ l_1 &= 1 \\ l_2 &= \lambda - k \end{aligned}$$

We now use the standard approach for solving linear homogeneous recurrence relations (see, e.g., [17]). The characteristic equation of the recurrence is:

$$x^2 - (\lambda + 1 - k)x + \lambda = 0$$

The roots of this equation are:

$$\begin{aligned} x_1 &= \frac{(\lambda + 1 - k) + \sqrt{(\lambda + 1 - k)^2 - 4\lambda}}{2} \\ x_2 &= \frac{(\lambda + 1 - k) - \sqrt{(\lambda + 1 - k)^2 - 4\lambda}}{2} \end{aligned}$$

We now show that the roots are irrational by verifying the following inequality.

$$\begin{aligned} (\lambda + 1 - k)^2 &< 4\lambda \\ \lambda^2 + 2(1 - k)\lambda + (1 - k)^2 &< 4\lambda \\ k^2 - 2k + 1 &< \lambda \cdot (2k + 2 - \lambda) \end{aligned}$$

Using the condition that $\lambda = (1 + \sqrt{k})^2 - \epsilon$ for some $\epsilon \in (0, 1)$, it suffices to verify that the following inequality holds for any such ϵ.

$$\begin{aligned} k^2 - 2k + 1 &< \left[(1 + \sqrt{k})^2 - \epsilon\right] \cdot \left[2k + 2 - \left((1 + \sqrt{k})^2 - \epsilon\right)\right] \\ k^2 - 2k + 1 &< \left[k + 2\sqrt{k} + 1 - \epsilon\right] \cdot \left[2k + 2 - k - 2\sqrt{k} - 1 + \epsilon\right] \\ k^2 - 2k + 1 &< \left[(k + 1) + (2\sqrt{k} - \epsilon)\right] \cdot \left[(k + 1) - (2\sqrt{k} - \epsilon)\right] \\ k^2 - 2k + 1 &< k^2 + 2k + 1 - 4k + 4\epsilon\sqrt{k} - \epsilon^2 \\ 0 &< 4\sqrt{k} - \epsilon \end{aligned}$$

Because $k \geq 1$, this inequality holds for any $\epsilon \in (0, 1)$. The two roots can then be represented as follows:

$$\begin{aligned} x_1 &= y + iz \\ x_2 &= y - iz \end{aligned}$$

where

$$\begin{aligned} y &= \frac{\lambda + 1 - k}{2} \\ z &= \frac{\sqrt{4\lambda - (\lambda + 1 - k)^2}}{2} \end{aligned}$$

Because the roots are distinct, the solution to the recurrence is of the form: $l_{j+1} = \delta_1 x_1^j + \delta_2 x_2^j$, where δ_1, δ_2 are constants that we now derive. The initial conditions give us the following equations:

$$\begin{aligned} 1 &= \delta_1 + \delta_2 \\ \lambda - k &= \delta_1 \cdot (y + iz) + \delta_2 \cdot (y - iz) \end{aligned}$$

Solving these equations yields:

$$\begin{aligned}\delta_1 &= \frac{1}{2} + \frac{\lambda - k - y}{2iz} \\ \delta_2 &= \frac{1}{2} - \frac{\lambda - k - y}{2iz}\end{aligned}$$

Because the pairs (x_1, x_2) and (δ_1, δ_2) are both complex conjugates with non-zero imaginary parts, we can represent the recurrence as follows for some $\alpha, \beta, \theta, \omega \neq 0$:

$$\begin{aligned}l_{j+1} &= \alpha e^{i\theta} \cdot (\beta e^{i\omega})^j + \alpha e^{-i\theta} \cdot (\beta e^{-i\omega})^j \\ l_{j+1} &= \alpha \cdot \beta^j [e^{i(\theta + j\omega)} + e^{-i(\theta+j\omega)}] \\ l_{j+1} &= \alpha \cdot \beta^j [\cos(\theta + j\omega) + i\sin(\theta + j\omega) + \\ &\quad \cos(-(\theta + j\omega)) + i\sin(-(\theta + j\omega))] \\ l_{j+1} &= 2 \cdot \alpha \cdot \beta^j \cos(\theta + j\omega)\end{aligned}$$

Because $\omega \neq 0$, it must be the case that $\cos(\theta + j\omega) < 0$ for some value of $j > 0$. Thus, since $\alpha, \beta > 0$, there must exist some $j > 0$ such that $l_{j+1} < 0$. Combined with the fact that $l_1 > 0$, this implies that there must exist an $m \geq 1$ such that $l_m < l_{m-1}$, completing the proof. □

We now present the proof of Theorem 6.

Theorem 17 *There does not exist a deterministic online mechanism that satisfies IC, IR, and NNP, and that achieves a competitive ratio less than* $(1 + \sqrt{k})^2 + 1$, *for any* $k > 1$.

Proof: Assume by contradiction that there exists a deterministic online mechanism Γ that satisfies IC, IR, and NNP, and that achieves a competitive ratio of $c = (1 + \sqrt{k})^2 + 1 - \epsilon$ for some $\epsilon > 0$. Since a competitive ratio of c implies a competitive ratio of $c + x$, for any $x > 0$, we assume without loss of generality that $\epsilon < 1$. First, we will construct a profile of agent types θ using an adversary argument. After possibly slightly perturbing θ to assure that a strictness property is satisfied, we will then use a more significant perturbation of θ to reach a contradiction.

We now construct the original profile θ. Pick an α such that $0 < \alpha < \epsilon$, and define $\delta = \frac{\alpha}{ck+3k}$. The adversary uses two sequences of jobs: minor and major. Minor jobs i are characterized by $l_i = \delta$, $v_i = k \cdot \delta$, and zero laxity. The first minor job is released at time 0, and $r_i = d_{i-1}$ for all $i > 1$. The sequence stops whenever Γ completes any job.

Major jobs also have zero laxity, but they have the smallest possible value ratio (that is, $v_i = l_i$). The lengths of the major jobs that may be released, starting with $i = 1$, are determined by the following recurrence relation.

$$\begin{aligned}l_{i+1} &= (c - 1 + \alpha) \cdot l_i - k \cdot \sum_{h=1}^{i} l_h \\ l_1 &= 1\end{aligned}$$

The bounds on α imply that $(1+\sqrt{k})^2 - 1 < c - 1 + \alpha < (1 + \sqrt{k})^2$, which allows us to apply Lemma 1. Let m be the smallest positive number such that $\frac{l_m + k \cdot \sum_{h=1}^{m-1} l_h}{l_m} > c - 1 + \alpha$.

The first major job has a release time of 0, and each major job $i > 1$ has a release time of $r_i = d_{i-1} - \delta$, just before the deadline of the previous job. The adversary releases major job $i \leq m$ if and only if each major job $j < i$ was executed continuously over the range $[r_i, r_{i+1}]$. No major job is released after job m.

In order to achieve the desired competitive ratio, Γ must complete some major job f, because $\Gamma_{offline}$ can always at least complete major job 1 (for a value of 1), and Γ can complete at most one minor job (for a value of $\frac{\alpha}{c+3} < \frac{1}{c}$). Also, in order for this job f to be released, the processor time preceding r_f can only be spent executing major jobs that are later abandoned. If $f < m$, then major job $f+1$ will be released and it will be the final major job. Γ cannot complete job $f+1$, because $r_f + l_f = d_f > r_{f+1}$. Therefore, θ consists of major jobs 1 through $f+1$ (or, f, if $f = m$), plus minor jobs from time 0 through time d_f.

We now possibly perturb θ slightly. By IR, we know that $v_f \geq p_f(\theta)$. Since we will later need this inequality to be strict, if $v_f = p_f(\theta)$, then change θ_f to θ'_f, which differs from θ_f only in that $v'_f = v_f + \delta$. By IC, job f must still be completed by Γ for the profile (θ'_f, θ_{-f}). If not, then by IR and NNP we know that $p_f(\theta'_f, \theta_{-f}) = 0$, and thus that $u_f(g(\theta'_f, \theta_{-f}), \theta'_f) = 0$. However, agent f could then increase its utility by falsely declaring the original type of θ_f, receiving a utility of: $u_f(g(\theta'_f, \theta_{-f}), \theta'_f) = v'_f - p_f(\theta) = \delta > 0$, violating IC. Furthermore, agent f must be charged the same amount (that is, $p_f(\theta'_f, \theta_{-f}) = p_f(\theta)$), due to a similar incentive compatibility argument. Thus, for the remainder of the proof, assume that $v_f > p_f(\theta)$.

We now use a more substantial perturbation of θ to complete the proof. If $f < m$, then define θ''_f to be identical to θ_f, except that $d''_f = d_{f+1} + l_f$, allowing job f to be completely executed after job $f+1$ is completed. If $f = m$, then instead set $d''_f = d_f + l_f$.

We now show that, for the profile $(\theta''_f, \theta_{-f})$, Γ must still execute job f continuously over the range $[r_f, r_f + l_f]$, thus preventing job $f+1$ from being completed. Assume by contradiction that this were not true. Then, at the original deadline of d_f, job f is not completed. Consider the possible profile $(\theta''_f, \theta_{-f}, \theta_x)$, which differs from the new profile only in the addition of a job x which has zero laxity, $r_x = d_f$, and $v_x = l_x = max(d''_f - d_f, (c+1) \cdot (l_f + l_{f+1}))$. Because this new profile is indistinguishable from $(\theta''_f, \theta_{-f})$ to Γ before time d_f, it must schedule jobs in the same way until d_f. Then, in order to achieve the desired competitive ratio, it must execute job x continuously until its deadline, which is by construction at least as late as the new deadline d''_f of job f. Thus, job f will not be completed, and, by IR and NNP, it must be the case that $p_f(\theta''_f, \theta_{-f}, \theta_x) = 0$ and $u_f(g(\theta''_f, \theta_{-f}, \theta_x), \theta''_f) = 0$. Using the fact that θ is indistinguishable from $(\theta_f, \theta_{-f}, \theta_x)$ up to time d_f, if agent f falsely declared his type to be the original θ_f, then its job would be completed by d_f and it would be charged $p_f(\theta)$. Its utility would then increase to $u_f(g(\theta_f, \theta_{-f}, \theta_x), \theta''_f) = v_f - p_f(\theta) > 0$, contradicting IC.

While Γ's execution must be identical for both (θ_f, θ_{-f}) and $(\theta''_f, \theta_{-f})$, $\Gamma_{offline}$ can take advantage of the change. If $f < m$, then Γ achieves a value of at most $l_f + \delta$ (the value of job f if it were perturbed), while $\Gamma_{offline}$ achieves a value of at least $k \cdot (\sum_{h=1}^{f} l_h - 2\delta) + l_{f+1} + l_f$ by executing minor jobs until r_{f+1}, followed by job $f+1$ and then job f (we subtract two δ's instead of one because the last minor job before r_{f+1} may have to be abandoned). Substituting in for l_{f+1}, the competitive ratio is then at least:

$$\begin{aligned}
& \frac{k \cdot (\sum_{h=1}^{f} l_h - 2\delta) + l_{f+1} + l_f}{l_f + \delta} \\
= \quad & \frac{k \cdot (\sum_{h=1}^{f} l_h) - 2k \cdot \delta + (c - 1 + \alpha) \cdot l_f - k \cdot (\sum_{h=1}^{f} l_h) + l_f}{l_f + \delta} \\
= \quad & \frac{c \cdot l_f + (\alpha \cdot l_f - 2k \cdot \delta)}{l_f + \delta} \\
\geq \quad & \frac{c \cdot l_f + ((ck + 3k)\delta - 2k \cdot \delta)}{l_f + \delta} \\
> \quad & c
\end{aligned}$$

If instead $f = m$, then Γ achieves a value of at most $l_m + \delta$, while $\Gamma_{offline}$ achieves a value of at least $k \cdot (\sum_{h=1}^{m} l_h - 2\delta) + l_m$ by completing minor jobs until $d_m = r_m + l_m$, and then completing job

m. The competitive ratio is then at least:

$$\begin{aligned}
& \frac{k \cdot (\sum_{h=1}^{m} l_h - 2\delta) + l_m}{l_m + \delta} \\
= \quad & \frac{k \cdot (\sum_{h=1}^{m-1} l_h) - 2k \cdot \delta + k l_m + l_m}{l_m + \delta} \\
> \quad & \frac{(c - 1 + \alpha) \cdot l_m - 2k \cdot \delta + k l_m}{l_m + \delta} \\
= \quad & \frac{(c + k - 1) \cdot l_m + (\alpha l_m - 2k \cdot \delta)}{l_m + \delta} \\
> \quad & c
\end{aligned}$$

□

Towards a General Theory of Non-Cooperative Computation*

(Extended Abstract)

Robert McGrew, Ryan Porter, and Yoav Shoham
Stanford University
{bmcgrew,rwporter,shoham}@cs.stanford.edu

Abstract

We generalize the framework of non-cooperative computation (NCC), recently introduced by Shoham and Tennenholtz, to apply to cryptographic situations. We consider functions whose inputs are held by separate, self-interested agents. We consider four components of each agent's utility function: (a) the wish to know the correct value of the function, (b) the wish to prevent others from knowing it, (c) the wish to prevent others from knowing one's own private input, and (d) the wish to know other agents' private inputs. We provide an exhaustive game theoretic analysis of all 24 possible lexicographic orderings among these four considerations, for the case of Boolean functions (mercifully, these 24 cases collapse to four). In each case we identify the class of functions for which there exists an incentive-compatible mechanism for computing the function. In this article we only consider the situation in which the inputs of different agents are probabilistically independent.

1 Introduction

In this paper we analyze when it is possible for a group of agents to compute a function of their privately known inputs when their own self-interests stand in the way. One motivation for studying this class of problems is cryptography. Consider, for example, the problem of secure function evaluation (SFE). In SFE, n agents each wish to compute a function of n inputs (where each agent i possesses input i), without revealing their private inputs. An increasingly clever series of solutions to SFE have been proposed (e.g., [1, 2]). But if these protocols are the answer, what exactly is the question? Like many other cryptographic problems, SFE has not been given a mathematical definition that includes the preferences of the agents. We hasten to add that this does not mean that the solutions are not clever or useful; they are. However, to prove that agents will actually follow a protocol, one needs a game-theoretic definition of the SFE problem. It turns out that the game theoretic analysis provides a slightly different perspective on (e.g.,) SFE; the paranoias of game theorists are more extreme than the traditional paranoias of cryptographers in some respects and less so in others. The difference between the two demands a more complete discussion than we have space for, and we discuss the issue in more depth in a companion paper.

In this paper we do not speak about cryptography *per se*, but rather about a general framework within which to think about cryptography and related phenomena. The framework is called *non-cooperative computing*, or NCC for short. The term was introduced by Shoham and Tennenholtz in [4], who adopt a narrower setting. The NCC framework of S&T is however too limited to account for (e.g.,) cryptography, and the goal of this paper is to extend it so it does.

We give the formal definitions in the next section, but let us describe the NCC framework intuitively. The setting includes n agents and an n-ary function f, such that agent i holds input i. Broadly speaking, all the agents want to compute f correctly, but in fact each agent has several

*This work was generously supported by DARPA grant F30602-00-2-0598 and by an NSF fellowship.

independent considerations. In this article we take agent i's utility function to depend on the following factors:

Correctness: i wishes to compute the function correctly.

Exclusivity: i wishes that the other agents do not compute the function correctly.

Privacy: i wishes that the other agents do not discover i's private input.

Voyeurism: i wishes to discover the private inputs of the other agents.

Of course, these considerations are often conflicting. They certainly conflict across agents – one agent's privacy conflicts with another agent's voyeurism. But they also conflict within a given agent – the wish to compute the function may propel the agent to disclose his private input, but his privacy concerns may prevent it. So the question is how to amalgamate these different considerations into one coherent preference function.

In this paper we consider lexicographic orderings. In the extended abstract, we analyze all 24 possible orderings of these considerations, while in the full paper we consider all possible orderings on all subsets of the considerations. In each case we ask for which functions f there exists a mechanism in the sense of mechanism design [3], such that in the game induced by the mechanism, it is a Bayes-Nash equilibrium for the agents to disclose their true inputs. Of course, to do that we must be explicit about the probability distribution from which the agents' inputs are drawn.

This is a good point at which to make clear the connection between our setting and the restricted NCC setting of S&T:

- S&T consider only *correctness* and *exclusivity* (and, in particular, only the ordering in which *correctness* precedes *exclusivity*).
- S&T consider both the case in which the inputs of the agents are independently distributed and the case in which they are correlated.
- S&T consider also a version of the setting in which agents are willing to mis-compute the function with a small probability, and another version in which agents can be offered money, in addition to their inherent informational incentives.

We not only consider *privacy* and *voyeurism* in addition to *correctness* and *exclusivity*, but also consider all 24 possible orderings among them (mercifully, in the Boolean case which we investigate they collapse to four equivalence classes), maintaining the property that all agents have the same ordering over the considerations. However, in this paper we do not investigate the case of correlated values, nor the probabilistic and monetary extensions. We leave those to future work.

There is one additional sense in which our treatment is more general. Consider for example the consideration of *correctness*, and three possible outcomes: in the first the agent believes the correct value with probability .6, in the second with probability .99, and in the third with probability 1. Holding all other considerations equal, how should the agent rank these outcomes? Clearly the third is preferred to the others, but what about those two? Here we have two versions; in one, the first two are equally desirable (in other words, any belief less than certainty is of no value), and in the other the second is preferred to the first. We call those two settings the *full information gain* setting and the *partial information gain* setting, respectively.

This means that rather than 24 cases we need to investigate, we have 48. But again, luck is on our side, and we will be able to investigate a small number of equivalence classes among these cases.

In the next section we give the precise definitions, and in the following sections we summarize our results. Several of the insights into our results are derived from the results obtained by S&T; we will try to indicate clearly when that is the case.

2 Formulation

2.1 Formal problem definition

As in NCC, let $N = \{1, 2, \ldots, n\}$ be a set of agents, and consider also a non-strategic center which will execute the protocol. We assume that each agent $1 \ldots n$ has a private and authenticated channel between itself and the center. Each agent has an input V_i drawn from the set B_i. We will use v_i (as shorthand for $V_i = v_i$) to represent a particular (but unspecified) value for V_i. The vector $v = (v_1, \ldots, v_n)$ consists of the types of all agents, while $v_{-i} = (v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$ is this same vector without the type of agent i ($v_{-i,j}$ simply extends this to removing two types). $P(V)$ is the joint probability over all players' inputs, which induces a $P_i(V_i)$ for each agent. Each agent knows his own type, but does not know the types of the other agents. Instead, the prior $P(V)$ (which we assume has full support – that is, $\forall v \; P(v) > 0$) is common knowledge among the agents and known by the mechanism designer. We further assume that the agent types are independent.

The commonly-known function that the agents are trying to compute is denoted by $f : B_1 \times \ldots \times B_N \rightarrow B_0$. Though the setting makes sense in the case of an arbitrary field, we restrict ourselves in this work to the case of Boolean functions over Boolean inputs ($B = B_i = \{0, 1\}$). We assume that all agents are *relevant* to the function in that they have at least some chance of affecting the outcome. Formally, this means that, for each agent i, $\exists v_i, v_{-i} \; f(v_i, v_{-i}) \neq f(\neg v_i, v_{-i})$.

2.2 The mechanism design problem

In general, a mechanism is a protocol specifying both the space of legal messages that the individual agents can send to the center and, based on these messages, what the center will return to each agent. We seek that, for all possible input values, all agents believe in the correct value of the function at the end of the protocol. Since dominant strategy implementation is not feasible in our setting, we are looking for a Bayes-Nash implementation.

A standard mechanism is a mapping from actions to outcomes. The setting of NCC is somewhat different from the standard mechanism design setting, however. In the case of NCC, an outcome is a complete set of belief states, but instead of mapping from actions to outcomes, the mechanism instead gives a signal to each player, who interprets it according to his *belief strategy.* Thus, a mechanism cannot enforce outcomes: it can only control the information embedded in its signal to each player. As we shall see, this will be sufficient for our purposes. A player's preferences over his and others' belief states are defined with respect to the correct inputs to and outputs of the function, as determined by the private types of the other players.

A priori, one could imagine arbitrarily complicated mechanisms in which the agents and the center iterate through many rounds of messages before converging on a result. However, following Shoham and Tennenholtz, we note that an extended revelation principle (extended from, e.g., [3]) allows us wlog to restrict our attention to mechanisms in which the agents truthfully declare an input to the center and accept the result returned to them.

Theorem 1 (Extended Revelation Principle) *If there exists a protocol for the center and the agents in which each agent computes the correct value of the function, then there exists a truthful, direct protocol in which each agent accepts the center's output and thereby computes the correct value of the function.*

Formally, a mechanism is a tuple $(S_1, \ldots, S_n, g_1, \ldots, g_n)$, consisting of, for each agent i, a strategy space S_i and a function g_i that determines the output returned to the agent. A strategy of agent i consists of the following tuple of functions: $(s_i : B \rightarrow \triangle B, b_i^f : B \times B \rightarrow \triangle B, b_i^1 : B \times B \rightarrow \triangle B, ..., b_i^n : B \times B \rightarrow \triangle B)$. The first maps an agent i's true type to a distribution over its declared type (which we will sometimes refer to as $\hat{v}_i$). The second maps i's true type v_i and the center's response $g_i(\hat{v})$ to i's beliefs about the output of the function f. The remaining functions map i's type and the center's response to i's beliefs about each agent j's private input. We shall henceforth refer to the tuple of belief functions, which together map to a complete belief state of agent i, as b_i, the agent's belief strategy. Agents may have other higher-order beliefs, but we can neglect these since they are not relevant to any agent's preferences.

The set of outcomes O is the set of distributions over the belief states of the agents about the input and output values; that is, $O = (\triangle B \times \triangle B^n)^n$. We wish to implement the social choice function W which always selects an outcome in which, for all agents i, $Pr(b_i^f(v_i, g_i(\hat{v})) = f(v)) = 1$. That is, in our desired outcome, each agent always computes the correct value of the function. In this paper, we restrict the range of $g_i : B^n \to B$ so that it returns to agent i a bit (to represent a possible output of the function) for each set of declared values $\hat{v}$. Since we wish every agent to always compute correctly, we can restrict the center's protocol to computing and returning the function $f(\hat{v})$ to each player (i.e. $g_i(\hat{v}) = f(\hat{v})$).

The agents' preferences are defined over the belief states which form the set of outcomes. We now give a more formal definition of the incentives of each agent, first for the *full information gain* setting.

Correctness: i wishes that $Pr(b_i^f(v_i, g_i(\hat{v})) = f(v)) = 1$.

Exclusivity: For each $j \neq i$, i wishes that $Pr(b_j^f(v_j, g_j(\hat{v})) = f(v)) \neq 1$.

Privacy: For each $j \neq i$, i wishes that $Pr(b_j^i(v_j, g_j(\hat{v})) = v_i) \neq 1$.

Voyeurism: For each $j \neq i$, i wishes that $Pr(b_i^j(v_i, g_i(\hat{v})) = v_j) = 1$.

In the *partial information gain* setting, agent valuations depend on more than whether or not a probability is equal to 1. Instead, agents attempt to maximize the entropy function, which for a distribution $Pr(X)$ over a Boolean variable X is defined as: $H(X) = -Pr(X = 0) \cdot log_2 Pr(X = 0) - Pr(X = 1) \cdot log_2 Pr(X = 1)$.

Because of the way in which we can reduce the space of mechanisms that we need to consider, we can restate our goal as follows. In this paper we characterize, for each possible ordering on the four incentives listed above, the set of functions f for which it is a Bayes-Nash equilibrium for each agent i to use a strategy $(s_i(v_i) = v_i, b_i^f(v_i, f(v)) = f(v), ...)$ – that is, always telling the truth and believing the output of the mechanism.

3 Full information gain setting

In this section we consider the *full information gain* setting, in which we assume that agents are only concerned with what they and the other agents know with certainty, as opposed to what they can know with some probability. We now characterize the set of functions that are NCC for each of 24 possible orderings of the incentives, which can be broken into four cases.

Before we begin, we review two definitions and a theorem from S&T [4] that will play an important role in our impossibility results. We say that f is *(locally) dominated* if there exists a type for some agent which determines the output of the function. Formally, the condition is that there exists an i and v_i such that $\forall v_{-i}, v'_{-i},\ f(v_i, v_{-i}) = f(v_i, v'_{-i})$. We say that a function f is *reversible* if there exists an agent i such that $\forall v_{-i},\ f(V_i = 0, v_{-i}) \neq f(V_i = 1, v_{-i})$, which means that for either input, agent i knows what the value of function would have been if he had submitted the other input.

Theorem 2 (Shoham and Tennenholtz) *When agents value* correctness *over* exclusivity*, and value no other consideration, a function is NCC if and only if it is non-reversible and non-dominated.*

We can restrict the class of functions to consider in the current setting by noting that any function which is not NCC in the S&T sense is not NCC for any ordering of the four incentives.

Theorem 3 *Any function that is reversible or dominated is not NCC.*

3.1 Exclusivity and correctness

We can tackle half of the orderings at once by considering the case in which all agents rank *exclusivity* over *correctness*. Not surprisingly, all is lost in this case.

Theorem 4 *If* exclusivity *is ranked over* correctness*, then no function is NCC.*

On the other hand, we find that the converse of Theorem 3 holds when *correctness* is ranked above all other factors.

Theorem 5 *If* correctness *is ranked over all other factors, then a function is NCC if and only if it is non-reversible and non-dominated.*

3.2 Privacy over correctness

We are now down to six cases, in which *correctness* must be ranked second or third, and *exclusivity* must be ranked below it. For the four of these cases in which *privacy* is ranked over *correctness*, the key concept is what we call a *privacy violation*, which occurs when an agent has an input for which there is a possible output that would allow the agent to determine another agent's input with certainty. Formally, we say that a *privacy violation* for agent i by agent j occurs whenever $\exists v_j, x, y, \forall v_{-j}\ (f(v_j, v_{-j}) = x) \Rightarrow (V_i = y)$.

Theorem 6 *If* privacy *is ranked over* correctness*, and both are ranked over* exclusivity*, then a function is NCC if and only if it is non-reversible, non-dominated, and has no privacy violations.*

It is interesting to note the relationship between privacy violations and what we call *conditional (local) domination.* We say that agent i *conditionally dominates* f given agent j if $\exists v_i, v_j, x\ (\forall v_{-i,j},\ f(v_i, v_j, v_{-i,j}) = x) \wedge (\exists v_{-i,j}\ f(\neg v_i, v_j, v_{-i,j}) \neq x)$. Using the terminology we defined earlier, conditional domination occurs when agent j can submit an input v_j such that agent i both dominates and is relevant to the output of the conditional function $f_{-j}(v_{-j}) = f(v_j, v_{-j})$.

Lemma 7 *: There exists a privacy violation for agent i by agent j if and only if agent i conditionally dominates f given j.*

3.3 Voyeurism first, correctness second

The final two cases to consider are those in which the first two considerations are *voyeurism* and *correctness*, in that order. If there exists an agent j who can obtain a greater amount of *voyeurism*, on expectation, from one of his possible inputs, then he will always choose to declare this input. Thus, a necessary condition for the function to be NCC is that the expected *voyeurism* be equal for $V_j = 0$ and $V_j = 1$. If this is the case, then *correctness* becomes paramount, and we again have the classic NCC condition.

Formally, define a new indicator function $violate(i, j, v_j, x)$ to be 1 if a privacy violation occurs for agent i by agent j, and v_j and x satisfy the condition for the violation to occur, and 0 otherwise. Now, we can formally give the condition for a *voyeurism tie.*

Theorem 8 *If all agents rank* voyeurism *first and* correctness *second, then, given a prior P(V), a function is NCC if and only if it is non-reversible and non-dominated and the following condition for a* voyeurism tie *holds for each agent j:*

$$\sum_{i \neq j} \sum_{v_{-j}} P(v_{-j}) \cdot violate\Big(i, j, V_j = 1, f(V_j = 1, v_{-j})\Big) = \sum_{i \neq j} \sum_{v_{-j}} P(v_{-j}) \cdot violate\Big(i, j, V_j = 0, f(V_j = 0, v_{-j})\Big)$$

For the common prior $P(0) = \frac{1}{2}$, an example of a function for which there is a voyeurism tie in the presence of privacy violations is the *unanimity* function, in which $f(v) = 1$ if and only if the inputs of all agents are identical.

Finally, note that the space of functions which are NCC in these two cases is a superset of the functions which are NCC in the cases of the previous subsection (*privacy* over *correctness*), since a complete lack of privacy violations trivially induces a voyeurism tie.

4 Partial information gain setting

Now we consider the *partial information gain* setting, in which agents value increased information about a factor. For this setting, we see that the results are unchanged for many of the possible lexicographic orderings, but are different for several interesting cases.

4.1 Unchanged results

The first three theorems from the *full information gain* setting carry over exactly to this setting.

Theorem 9 *In the* partial information gain *setting, any function that is reversible or dominated is not NCC.*

Theorem 10 *In the* partial information gain *setting, if agents rank* exclusivity *over all other factors, then no function is NCC.*

Theorem 11 *In the* partial information gain *setting, if agents rank* correctness *over all other factors, then a function is NCC if and only if it is non-reversible and non-dominated.*

4.2 Privacy over correctness

For the cases in which *privacy* is ranked above *correctness*, the condition for non-cooperative computability becomes more stringent, and it now depends on the form of the prior.

First, we need to update the definition of a privacy violation, because it now occurs whenever an agent's posterior distribution over another agent's input differs at all from the prior distribution. We say that a *partial privacy violation* of agent i by agent j occurs if $\exists v_i, x,\ Pr(V_j|v_i, f(v) = x) \neq P(V_j)$.

Theorem 12 *In the* partial information gain *setting, if agents rank* privacy *over* correctness*, then, given a prior $P(V)$, a function is NCC if and only if it is non-reversible and non-dominated and there are no partial privacy violations.*

The absence of partial privacy violations can also be formulated by the following condition, which, in words, requires that no pair of inputs provide more information about the output than any other pair.

Lemma 13 *A function has no partial privacy violations if and only if satisfies the following condition:*

$$\exists c,\ \forall i, j, v_i, v_j,\ Pr(f(v) = 0|v_i, v_j) = c$$

A (relatively) simple function that satisfies this condition is: $f(v) = parity(v_1, v_2, v_3) \wedge parity(v_4, v_5, v_6)$, with a common prior of $P(0) = \frac{1}{2}$. There also exist privacy-preserving functions that treat each agent's input symmetrically. One example, for $N = 7$ and the common prior $P(0) = \frac{1}{2}$, is the function $f(v)$ that returns 1 if and only if the number of agents i for which $v_i = 0$ is 1, 2, 4, or 5.

4.3 Voyeurism first, correctness second

The last two cases to consider are those in which agents rank *voyeurism* first and *correctness* second, leaving *privacy* as their third or fourth priority.

In order to calculate the expected entropy that agent j has agent i's input after learning the output of the function, we can use the following expression: $Pr(v_i|f(v) = x, v_j) = \frac{Pr(f(v)=x|v_i,v_j)\cdot P(v_i)}{Pr(f(v)=x|v_j)}$.

For the desired equilibrium to hold, it must be the case that for each agent i the expected partial *voyeurism* is the same for both possible values of V_i.

Theorem 14 *In the* partial information gain *setting, if agents rank* voyeurism *first and* correctness *second, then, given a prior P(V), a function is NCC if and only if it is non-reversible and non-dominated and the following condition holds for each agent j:*

$$\sum_{i \neq j} E_x[H(V_i|V_j = 1, f(v) = x)] \neq \sum_{i \neq j} E_x[H(V_i|V_j = 0, f(v) = x)]$$

Note that, for the common prior $P(0) = \frac{1}{2}$, the *unanimity* function still induces a voyeurism tie, as it did for the same two orderings of the *full information gain* setting.

5 Conclusion

In this paper, we have considered a class of incentive structures for agents and a class of mechanisms, and characterized the sets of functions which are computable by agents which are similarly self-interested. A summary of our results lends itself to a decision tree, as shown in Figure 1.

We view these results as laying the groundwork for a consideration of a wide variety of both theoretical concerns and practical problems. In the introduction, we discussed the cryptographic problem of secure function evaluation. Determining whether these cryptographic protocols will lead to successful computation requires considering not only deviations from the protocol given agent inputs (which is the extent of the analysis in most papers in this field), but also whether the protocol is incentive compatible. Using the impossibility results stated above, we can focus our efforts on designing protocols for functions which are non-cooperatively computable.

In addition, we can extend our formulation along several dimensions. For example, if we allow the mechanism to return to an agent the inputs of other agents, in addition to the output of the function, then *voyeurism* no longer prevents a function from being NCC. Intuitively, the mechanism will expose inputs in a way that always creates a voyeurism tie. While similar solutions cannot overcome *privacy* or *exclusivity*, other extensions, including correlation between agent inputs and the possibility of monetary payments, further expand the space of functions that are NCC. Finally, the analysis and types of results we obtained are not limited to Boolean functions and lexicographic utility functions, and we regard extending this analysis to more general fields as a promising line of future research.

References

[1] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. STOC'88.

[2] Shafi Goldwasser. Multi party computations: past and present. Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing, 1989. ACM.

[3] A. Mas-Collel, W. Whinston, and J. Green. Microeconomic Theory. 1995.

[4] Yoav Shoham and Moshe Tennenholtz. Non-Cooperative Evaluation of Logical Formulas: The Propositional Case. Under review. 2003.

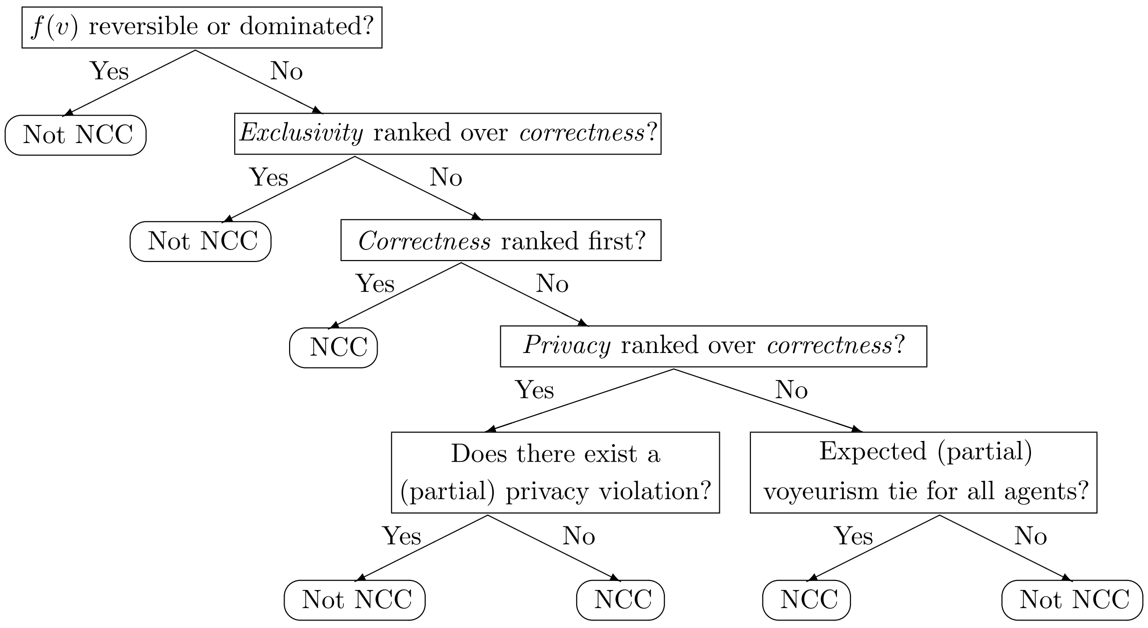

$f(v)$ reversible or dominated?
Yes
No
Not NCC
Exclusivity ranked over *correctness*?
Yes
No
Not NCC
Correctness ranked first?
Yes
No
NCC
Privacy ranked over *correctness*?
Yes
No
Does there exist a (partial) privacy violation?
Expected (partial) voyeurism tie for all agents?
Yes
No
Yes
No
Not NCC
NCC
NCC
Not NCC

Figure 1: A decision tree which summarizes the conditions for a function and prior to be NCC. The conditions are the same for the two settings we consider, except for the bottom two decision boxes, in which "(partial)" refers to the updated conditions for the *partial information gain* setting.

Using Contracts to In uence the Outcome of a Game*

Robert McGrew and **Yoav Shoham**
Computer Science Department
Stanford University
Stanford, CA 94305
{bmcgrew, shoham}@cs.stanford.edu

Abstract

We consider how much in uence a center can exert on a game if its only power is to propose contracts to the agents before the original game, and enforce the contracts after the game if all agents sign it. Modelling the situation as an extensive-form game, we note that the outcomes that are enforceable are precisely those in which the payoff to each agent is higher than its payoff in at least one of the Nash equilibria of the original game. We then show that these outcomes can still be achieved without any effort actually expended by the center: We propose a mechanism in which the center does not monitor the game, and the contracts are written so that in equilibrium all agents sign and obey the contract, with no need for center intervention.

Introduction

There has been much interest in AI in mechanism design, the area of game theory devoted to designing protocols for self-interested agents. In the literature (Mas-Colell, Whinston, & Green 1995) it is generally assumed that the mechanism designer has complete freedom in designing the rules of the game. Yet the world is full of strategic situations with rules that already exist and cannot be changed arbitrarily. Recent work on *k-implementation* (Monderer & Tennenholtz 2003) restricts the capabilities of the mechanism designer in a particular way – it can add to any given cell in the payoff matrix, but it cannot subtract. (The interesting results in that line of work concern cases in which, despite that addition, the cost to the center in equilibrium is zero.) The opposite of this setting would be one in which the center can impose nes, rather than bonuses. This in and of itself is not interesting, because with suf ci ently large nes any outcome can be enforced. However, suppose the mechanism cannot unilaterally impose nes, but it can do so in the context of a signed contract. Speci cally , we consider the following class of mechanisms. Given a game G, the center can:

1. Propose a contract before G is played. This contract speci es a particular outcome, that is, a unique action for each agent, and a penalty for deviating from it.
2. Collect signatures on the contract and make it common knowledge who signed.
3. Monitor the players' actions during the execution of G.
4. If the contract was signed by all agents, ne anyone who deviated from it as speci ed by the contract.

Our setting is reminiscent of the work on social laws and conventions (Shoham & Tennenholtz 1997). There too the center can offer a *social convention* to the players, where each player agrees to a particular outcome so long as the other players play their part. The difference is that in that work it is assumed that, once all agents agree, the center has the power to enforce that outcome. Here we assume that players still have the freedom to choose whether or not to honor the agreement; the challenge is to design a mechanism such that, in equilibrium, they will.

The technical results of this paper will refer to games of complete information, but for intuition consider the example of online auctions, such as those conducted by eBay. Consider the complete game being played, including the decision after the close of the auction by the seller of whether to deliver the good and by the buyer of whether to send payment. Straightforward analysis shows that that the equilibrium is for neither to keep his promise, and the experience with fraud on eBay (Hafner 2004) demonstrates that the problem is not merely theoretical. It would be in eBay's interest to nd a way to enable its customers to bind themselves to their promises.

Our rst interest will be to characterize the achievable outcomes: What outcomes may the center suggest, with associated penalties, that the agents will accept? Our rst result will be an observation that the center's power is quite broad: Any outcome will be accepted when accompanied by appropriate nes, so long as the payoffs of each agent in that outcome are better than that player's payoffs in *some* equilibrium of the original game.

Although the center can achieve almost any outcome, we note that the helpful center expends a large amount of effort to do so: suggesting an outcome, collecting signatures, observing the game, and enforcing the contracts. If this procedure happens not just for one game, but for hundreds or thousands per day, the center may wish to nd a way to avoid this burden while still achieving the same effect.

*This work was supported in part by the National Science Foundation under ITR IIS-0205633 and by This work was supported in part by the DARPA grant F30602-00-2-0598.

The bulk of this paper concerns ways in which this reduction in effort can be achieved. We continue to assume that the center still needs to propose a contract. We also simply assume that it does not monitor the game. Nor does it participate in the signing phase; the agents do that among themselves using a broadcast channel. While we might imagine that the players could simply broadcast their signatures, this protocol allows a single player to learn the others' signatures and threaten them with nes. Nonetheless, we can construct a more complicated protocol - using a second stage of contracts - which does not require the center's participation. The only phase in which the center's protocol requires it to get involved under some conditions is the enforcement stage. However, our goal will be to devise contracts so that, *in equilibrium*, at this stage too the center sits idle. Our results here will be as follows. If the game play is *veri able* (if the center can discover after the fact how the game played out), we can achieve all of the outcomes achievable by a fully engaged center. If the game is not veri able then we can still achieve all previously achievable outcomes with some contract, but that contract might allow equilibria with additional outcomes.

The rest of the paper is organized as follows: we rst formally de ne our setting. Then we characterize the set of outcomes which are achievable with a busy center using contracts in this game. Finally, we lighten the load on the center rst in the enforcement stage, then in the signature exchange stage.

Formal Setting

The strategic situation the center wishes to in uence can be characterized as a *strategic-form game* with consequences in O: $G = \langle N, A, O, g, V \rangle$. (We roughly follow the notation of (Osborne & Rubinstein 1994).) Here N is the set of players $\{1...n\}$. $A = A_1 \times A_2 \times ... \times A_n$, where A_i is the set of actions which can be taken by an individual agent. We will use a_i to refer to an action of i in G and a_{-i} to refer to the vector of actions of all other players. O is the space of outcomes; $g : A \to O$ determines the outcome after an action pro le. We identify each outcome $o_{(a_i,a_{-i})}$ with a distinct action pro le (a_i, a_{-i}), and assume $g(a_i, a_{-i}) = o_{(a_i,a_{-i})}$. $V = V_1 \times V_2 \times ... \times V_n$, where $V_i : A \to \mathbb{R}$ is the pay-off function for player i.

Before this strategic situation G occurs, the helpful center suggests a contract to the players. This contract speci es the outcome o suggested by the center and what actions h the center will take in response to different action pro les of the players. The center will not enforce this contract unless it is signed by all players. This contract de nes the center's protocol in the enforcement stage H, as described below. We will denote a contract that describes a particular center protocol h as c_h.

Now we will describe the stages of the game, as initially formulated. We will adjust this formulation in later sections so that the center does no work in equilibrium.

Signature Exchange Stage F Each player who assents sends his signature on the contract to the center, who collects them. The center noti es all players of the identities of the signers. At the end of this stage, it is common knowledge whether or not the contract will be enforced.

Execution Stage G The players play the game G. Each player may take his action a_i to achieve o or he may not. The center observes the actions taken by the players.

Enforcement Stage H The center takes the actions speci ed in the contract in response to the actions he observed.

The outcomes are a consequence of the execution stage, but the only way the center can affect the players' actions in the is by ning them in the enforcement stage.

The extended game which arises from playing the stage games one after another we denote by $X = F \cdot G \cdot H$. Together, these de ne an *extensive-form game with simultaneous moves*. In general, an extensive-form game X can be de ned as $X = \langle N, \Omega, A_\omega, P, U \rangle$, where N is again the players, Ω is the set of histories of actions taken, A_ω is the set of actions for all players that can be taken after history ω, $P : \Omega \to 2^N$ is the player function that de nes which players get to move after a given history, and U is the utility function of players in the entire game.

In our particular setting, the history ω is just the set of actions taken in each stage game played so far, A_ω is the set of actions possible in each stage game following history ω, P is the set of all players (all players move simultaneously in each stage), and U is the (undiscounted) sum of the utilities of each stage game. We denote the subgame of $X = \langle N, \Omega_{|\omega}, A_{|\omega}, P_{|\omega}, U_{|\omega} \rangle$ that arises after history ω by $\Gamma(\omega)$, which simply refers to the play of the remaining stage games following the actions taken in ω. In later sections, we will refer to the strategy space of stage F as Γ^F, of stage G as Γ^G, and of stage H as Γ^H.

A *pure strategy* σ_i for player i in a strategic-form game corresponds to the choice of a single action $\sigma_i \in A_i$. A *mixed strategy* corresponds to the choice of a distribution over actions: $\sigma_i \in \Delta A_i$. A pure strategy in an extensive form game is de ned as $\sigma_i : \Omega \to A_\omega$; a mixed strategy is de ned accordingly. If σ_i is a strategy in X, then the strategy $\sigma_{i|\omega} : \Omega_{|\omega} \to A_{|\omega}$ induced by σ_i in the subgame $\Gamma(\omega)$ is $\sigma_{i|\omega}(\omega') = \sigma_i(\omega, \omega')$. Since it is unobservable whether a player has played a particular mixed strategy (only the realization of that strategy is observed), we will henceforth concentrate on enforcing outcomes that are the consequence of pure strategy pro les.

Our chosen solution concept will be subgame perfect equilibrium. To de ne this, we must r st de n e a *Nash equilibrium*: a pro le of strategies σ is a Nash equilibrium in a game if $\forall i \in N, \sigma'_i \in \Delta A_i : U_i(\sigma_i, \sigma_{-i}) \geq U_i(\sigma'_i, \sigma_{-i})$. A pro le of strategies σ in an extensive form game is a *subgame perfect equilibrium* if for every $\omega \in \Omega$, $\sigma_{|\omega}$ is a Nash equilibrium of the subgame $\Gamma(\omega)$. A subgame perfect equilibrium is resistant to deviations by players even in subgames off the equilibrium path.

The Power of a Helpful Center

We wish to characterize the power of a helpful center without any resource limitations. In this section, the center is limited only by the voluntary consent required from all agents and by its lack of desire to spend its own money. Speci cally , we assume that it collects the signatures in F itself and that it monitors the players' actions in G. In later sections we will relax each of these two assumptions.

First, we must precisely de ne the game which is being played. We model the signature exchange stage as a game form F with players N and action space $\Gamma^F = \{0,1\}^n$. The player i assents to the contract if $\gamma_i^F \in \Gamma_i^F = 1$. Since the center broadcasts the identities of the signers, each player's action is common knowledge. The execution game G thus has an extended action space $\Gamma^G : \{0,1\}^n \to A$ in which players decide to take action based on the consequences of the signature exchange stage. In the initial formulation, the enforcement stage H requires no action on the part of the players, but only of the center. The center's protocol h sets the payoff function of the enforcement stage. The center observes the signatures it receives and the actions chosen by the players and chooses to ne or reward players. Formally, $h = h_1 \times h_2 \times ... \times h_n$ and $h_i : \{0,1\}^n \times O \to \mathbb{R}$.

We de ne the payoff function $U_i : \Gamma \to \mathbb{R}$ for each player in the extended game X given actions $v \in \{0,1\}^n$ and $(a_i, a_{-i}) \in A$ as $U_i(v, a_i, a_{-i}) = V(g(a_i, a_{-i})) + h_i(v, g(a_i, a_{-i}))$. Thus each player has a quasi-linear utility function over the outcome determined in G and the money taken or given by the center according to h.

We say that the center's protocol h is *voluntary* if the center neither n es nor rewards players if the contract is not signed by every player: for all $v \neq 1^n \in \{0,1\}^n$ and for all $o \in O$, it is the case that $h_i(v, o) = 0$. We say that h is *frugal* if the center never spends its own money: for all $v \in \{0,1\}^n$ and all $o \in O$, it is the case that $\sum_{i \in N} h_i(v, o) \leq 0$. As these capture the limitations on the helpful center in our setting, we will henceforth limit h to be frugal and voluntary.

We rst wish to characterize what outcomes can occur in a subgame-perfect equilibrium of the extended game X. The outcome depends on two things: the contract c_h suggested by the center and the strategies of the players in X. We wish to nd contracts to which the players will agree that ensure that our chosen outcome is played.

In order to characterize the space of possible outcomes which can be enforced, we must de ne the notion of a *punishment equilibrium*. ρ^i is a punishment equilibrium for i if ρ^i is the Nash equilibrium of G with minimal payoffs for i among all (mixed) Nash equilibria of G.

Theorem 1 *Let ρ^i be the punishment equilibrium for i. For all $o_{(a_i,a_{-i})}$, if $V_i(a_i, a_{-i}) \geq V_i(\rho^i)$, then there exists a voluntary and frugal center protocol h and a subgame perfect equilibrium π^* in which all players agree to c_h and play (a_i, a_{-i}), and in no subgame perfect equilibrium do players agree to c_h and then fail to play (a_i, a_{-i}). Furthermore, for all i, $U_i(\pi^*) = V_i(a_i, a_{-i})$. If $V_i(a_i, a_{-i}) < V_i(\rho^i)$, then there is no subgame perfect equilibrium in which (a_i, a_{-i}) is played.*

Proof: First, suppose $V_i(a_i, a_{-i}) < V_i(\rho^i)$. Since player i will get at least $V_i(\rho^i)$ in any subgame perfect equilibrium without nes, i can pro t by withholding his assent. As (a_i, a_{-i}) cannot be a Nash equilibrium by assumption and no nes are assessed in H, there can be no subgame perfect equilibrium in which (a_i, a_{-i}) is played.

Second, suppose $V_i(a_i, a_{-i}) \geq V_i(\rho^i)$. We choose $h_i(a_i) = 0$ and $h_i(a_i' \neq a_i) = -M$. If we choose M so that for all i, a_i', and a_{-i}', it is the case that $M > V_i(a_i', a_{-i}') - V_i(a_i, a_{-i})$, then (a_i, a_{-i}) will be the only subgame perfect equilibrium of the subgame $G \cdot H$, supposing all players agree to c_h. We also require that all players assent in F. If any player does not assent, all players coordinate on his punishment equilibrium ρ^i in G. If more than one player fails to assent, we break ties arbitrarily to see which ρ^i is played. No matter which player fails to assent, ρ^i will be a subgame perfect equilibrium of $G \cdot H$, since the center will not assess nes. Thus i will not pro t by withholding his assent. $\square$

Thus we show that, with a fully engaged center that takes part in the protocol and monitors the players' actions, we can achieve any payoffs for the players which are at least as good for every player as some Nash equilibrium of G. Furthermore, once a contract for o is mutually signed, the unique subgame perfect equilibrium achieves o.

We notice that, already, the center takes no action in H in equilibrium. Yet as the center takes action in every other stage, we shall consider how to lighten the load on the center.

Removing the Center From the Enforcement Stage

In this section, we will drop the assumption that the center does not monitor the players' actions in the execution stage G. Instead, we assume that actions and outcomes are common knowledge among the players but are not observed by the center. The center must therefore encourage the players to tell him if there has been a deviation. We will distinguish two cases. In the *veri able* case, the center can verify that a particular player played a given action if he chooses to do so once the game G has been played. Speci cally , we require that the center be able to verify, for each player i, whether i played the correct action a_i or some other action $a_i' \neq a_i$. The center saves effort by not paying attention to G; we merely require that he can determine the truth after the fact, if necessary. In the *unveri able* case, the center has no information about players' actions whatsoever.

Because we now require the center to be noti ed by the players of deviations, the enforcement games we now consider will be of the following form: rst, the players observe the outcome and send messages to the center. The center publishes any messages he receives to all players. The players then have the chance to respond to the center's messages. This repeats for some number of rounds. Finally, the center makes monetary transfers between the players based on the messages sent.

For our purposes, this full generality is not needed. Our enforcement stage H is a single-round stage game where each player chooses whether or not to *complain* about other players by sending their names to the center, and the center chooses a ne to impose on each player: $H = \langle N, \Gamma^H, \mathbb{R}^n, h \rangle$. $\gamma_i^H \in \Gamma_n^H : O \to 2^N$ speci es which complaints player i will send to the center after each outcome. As before, h is the center's protocol which maps outcomes and complaints received to monetary consequences in $\mathbb{R}^n$. The center may make payments based on the outcome (if he can verify it), the identities of the complainers, and the target of their complaints. In the veri able case, $h : O \times (2^N)^n \rightarrow \mathbb{R}^n$, while in the unveri able case, $h : (2^N)^n \rightarrow \mathbb{R}^n$.

Now that we have speci ed an enforcement game, we wish to characterize the set of outcomes obtainable thereby in the extended game corresponding to this enforcement game.

We de ne a protocol h_o for the center, which will induce an equilibrium under which the center takes no action in the enforcement stage. Let M and m be a large and small amount of money, respectively. In h_o, the center punishes each player who deviated by a large-enough amount M, but also rewards each player who sent in a correct complaint by m for each correct complaint. The center also punishes any player who sent in an incorrect complaint by m. The contract that speci es center protocol h_o we call c_{h_o}.

Theorem 2 (Contracts for Veri able Games) *Let G be a game with veri able consequences in O and let* $o_{(a_i,a_{-i})} \in$ *O be the desired outcome. Assume that the center has suggested contract* c_{h_o} *de ned above and consider the subgame* $G \cdot H$ *that follows unanimous agreement to this contract. Then there is a strategy pro le* π^* *such that* π^* *is the unique subgame perfect equilibrium of* $G \cdot H$*,* $o_{(a_i,a_{-i})}$ *is the equilibrium outcome of* π^**, and* π^* *has payoffs* $V(a_i, a_{-i})$*. The center takes no action if* π^* *is played.*

[Proof omitted.]

We now consider the unveri able case. As before, we rst de ne a particular center protocol h'_o. In h'_o, the center punishes the target of each complaint by a large-enough amount M, but does not reward or punish players for complaints. After all, the center cannot distinguish valid complaints from invalid ones. The contract that speci es center protocol h'_o we call $c_{h'_o}$.

Theorem 3 (Contracts for Unveri able Games) *Let G be a game with unveri able consequences in O, and let* $o_{(a_i,a_{-i})} \in O$ *be the desired outcome. Assume that the center has suggested contract* $c_{h'_o}$ *and consider the subgame* $G \cdot H$ *that follows unanimous agreement to this contract. Then there is a strategy pro le* $\pi^{*\prime}$ *such that* $\pi^{*\prime}$ *is a subgame perfect equilibrium of* $G \cdot H$*,* $o_{(a_i,a_{-i})}$ *is the equilibrium outcome of* $\pi^{*\prime}$*, and* $\pi^{*\prime}$ *has payoffs* $V(a_i, a_{-i})$*. The center takes no action if* $\pi^{*\prime}$ *is played.*

[Proof omitted.]

We have seen that even without veri ability , it is possible to achieve almost any outcome in equilibrium. Unfortunately, these equilibria are no longer unique. As we shall see, in the unveri able case, a given signed contract may have many possible equilibrium outcomes rather than just the intended one.

Given a game G, de ne the *shortfall* s_i^σ of pure-strategy pro le $\sigma = (a_i, a_{-i})$ for i as $s_i^\sigma = \max_{a'_i} V_i(a'_i, a_{-i}) - V_i(a_i, a_{-i})$. The shortfall of i in σ is the amount i's payoffs would need to rise so that i would have no incentive to deviate from σ, all else held constant. We can see that there must be some equilibrium of the enforcement game in which an agent i is punished by at least $s_i^{(a_i,a-i)}$ whenever he deviates from his action a_i. Yet, in an unveri able game, there is nothing in the center's protocol which makes (a_i, a_{-i}) special. The players could just as well coordinate on this equilibrium in the enforcement game when the actions are not some other action (a'_i, a_{-i}). This implies that any enforcement scheme for the unveri able case will not in general have a unique outcome. Here we consider not only our chosen center protocol h'_o, but in fact any center protocol h in any form of enforcement game H.

Theorem 4 (Spurious Equilibria) *Consider an unveri able enforcement game with a frugal and voluntary h under which* $G \cdot H$ *has a subgame-perfect equilibrium* π *in which the center does no work, where* (a_i, a_{-i}) *is the strategy pro le that* π *plays in G. Then if* σ' *is a pure strategy pro le of G and* $\forall i : s_i^{\sigma'} \leq s_i^{(a_i,a-i)}$*, then there exists a subgame perfect equilibrium* π' *of* $G \cdot H$ *such that* σ' *is the strategy pro le that* π' *plays in G.*

[Proof omitted.]

A consequence of this theorem is that any Nash equilibrium of G can be played in $F \cdot G \cdot H$ regardless of the contract signed.

Corollary 5 (No Deletion) *If* σ *is a pure or mixed Nash equilibrium in the unveri able game G, then, for any frugal and voluntary center protocol h that has a subgame perfect equilibrium* π *where the center does no work, there is a subgame perfect equilibrium* π' *of* $G \cdot H$ *such that* σ *is the strategy pro le that* π' *plays in G.*

[Proof omitted.]

Thus, if the center cannot verify the players' actions, he cannot in general enforce any outcomes uniquely. After signing the contracts, the players might arrive at an outcome different from the one the center suggested. In a real-world setting, this would substantially weaken the case that the players should sign the contract.

We have shown that a helpful center who neither monitors the player's actions nor nes any player in equilibrium can enforce every outcome that a fully engaged center can enforce with more burdensome contracts. In an unveri able game, however, the center must generally accept spurious equilibria. Our next task is to remove the center from the signature exchange stage.

Exchanging Signatures Without The Center

Under the original contract, the center collected signatures on the contract c_h and enforced the contract if every player signed. We now show how the players can exchange signatures on the contract by use of a broadcast channel without requiring any action from the center in equilibrium. In this, our goal is similar to the goal of *optimistic signature exchange* (Garay & MacKenzie 1999), but with rational actors instead of computationally-bounded ones.

If players may communicate without being observed by others, F would be a game of imperfect information. As these games are dif cult to analyze and generally admit of many solutions, we require the players to use a broadcast channel, on which all messages sent are common knowledge.

When the center no longer monitors the signature exchange stage, he no longer knows in the enforcement stage H whether the contracts have been signed or not. Therefore, we now require that each complaint sent to the center in the enforcement stage H include a fully signed copy of the contract.

The Naive Broadcast Protocol

We might hope that the signature collection service performed by the center was super uous: that we will achieve the same results if we simply require players to broadcast their agreement or disagreement. Unfortunately, this will not be so. Consider the naive broadcast protocol where all players simultaneously broadcast their signatures. Let us formally de ne F to be the one-round stage game $F = \langle N, \Gamma^F, S, f\rangle$, where N is the set of players and $\Gamma^F = \{0,1\}^n$, where 0 represents the decision not to broadcast one's signature, while 1 represents the decision to do so. $S = (\{0,1\}^n)^n$ is the set of outcomes of the game. Each outcome speci es the set of signatures (represented by $\{0,1\}^n$) possessed by each player in N. $f : \Gamma^F \rightarrow S$ is the outcome function of F: each player knows his own signature and every signature which is broadcast.

The complete set of signatures is thus common knowledge if and only if every player chooses to broadcast his signature. Consider what occurs if exactly one player i fails to reveal his signature: i has received all the signatures of the other players, and he can produce his own. Thus i is the only player to possess all signatures on the contract, and this fact is common knowledge among the players. The center, on the other hand, cannot distinguish this case from the case where all players know all signatures, but only i chooses to complain. Therefore i is able to unilaterally enforce the contract, unlike in the original formulation.

Recall that, in H, every message sent by one player to the center is broadcast to all other players. Thus, once one player has sent a complaint about another (which includes a fully signed contract), every player will know all signatures on c_h and be able to complain. A player who deviates in F cannot choose to punish other players while remaining unscathed himself, but he can choose unilaterally whether or not to enforce the contract. Unfortunately, this power implies that our previously speci ed equilibrium for the extended game $F \cdot G \cdot H$ is no longer an equilibrium.

The equilibrium for $F{\cdot}G{\cdot}H$ discussed above requires that, if i fails to reveal his signature on the contract, all players coordinate on i's punishment equilibrium. Consider the case, for instance, where the punishment equilibrium for some i is (a_i, a'_{-i}), where a_i is the action i is contractually obliged to play. Suppose i alone fails to reveal his signature and all players play i's punishment equilibrium (a_i, a'_{-i}). In stage H, then, i will pro t by choosing to enforce the contract: the center will punish the other players and reward i. Knowing this, the other players will not in general wish to play their part of the punishment equilibrium, so our previous strategy fails.

The Pre-Contract Protocol

Although the naive broadcast protocol did not allow us to guarantee all the payoffs we wanted, we shall see that we can use a more complicated signature exchange stage F to ensure that either each player receives all signatures on c_h, or no players receive all signatures on c_h. Our exchange scheme is modelled on the contracts mechanism of the rest of the paper: we will add a *pre-contract* $\hat{c}_h$ that the players will sign before signing c_h. This contract authorizes the center to ne players who do not reveal their signature on c_h. Surprisingly, this does not lead to in nite regress: this one pre-contract is suf cient to allow for signature exchange.

We will divide F itself into stages: a miniature contract exchange stage $\hat{F}$, a miniature execution stage $\hat{G}$, and a miniature enforcement stage $\hat{H}$. The players will bind themselves in contract $\hat{c}_h$ to reveal their signatures on the contract c_h in such a way that, if they fail to reveal them, they can be ned by the center. We will allow them, however, to recoup that ne by revealing their signatures on c_h to the center and all players after the fact. The after-the-fact alteration of the outcome allows us to use the naive broadcast protocol for $\hat{F}$ where we could not use it for F.

Formally, let the signature stage $F = \hat{F} \cdot \hat{G} \cdot \hat{H}$. Let us call the contract signed in $\hat{F}$ the pre-contract $\hat{c}_h$, which binds players to release their signatures on the real contract c_h. $\hat{F}$ is the naive broadcast protocol de ned above as the stage game $\hat{F} = \langle N, \Gamma^{\hat{F}}, S, \hat{f}\rangle$. S is the set of signatures on $\hat{c}_h$ that each player knows, and $\Gamma^{\hat{F}}$ is each player's choice to broadcast or not broadcast his signature on $\hat{c}_h$. $\hat{G}$ is also the naive broadcast protocol de ned above for the signatures on c_h: $\hat{G} = \langle N, \Gamma^{\hat{G}}, \Phi, \hat{g}\rangle$, with Φ the set of known signatures on c_h, and $\Gamma^{\hat{G}} : S \rightarrow 0, 1^n$ the decision of the players to broadcast their signatures on c_h given what signatures each player knows on $\hat{c}_h$.

$\hat{H} = \langle N, \Gamma^{\hat{H}}, \mathbb{R}^n, \hat{h}\rangle$ is a miniature enforcement stage that is substantially different from the enforcement stage H. $\hat{H}$ has two rounds. In the rst round, a player i is allowed to complain to the center that he has not received some signature on c_h. To do so, i must submit the contract $\hat{c}_h$, all signatures on $\hat{c}_h$, and i's signature on c_h. When the center rebroadcasts this message to all players, both the signatures on $\hat{c}_h$ and i's signature on c_h become known to all players. In the second round, each player who did not complain in the rst round is given a chance to complain. $\Gamma_i^{\hat{H}}$ simply characterizes whether i will complain to the center after each history.

We now state our chosen center protocol $\hat{h}$ in $\hat{H}$. If the center received a complaint in the rst round, then the center nes all players who did not complain in either the rst or the second round by a large-enough amount M. If the center does not receive any complaints, he does not ne any players. Note that if all players complain, all signatures on c_h become common knowledge and no nes are assessed.

We now specify the strategy π^* we expect the agents to play in F. In $\hat{F}$, each player rst reveals his signature on $\hat{c}_h$. In $\hat{G}$, he will reveal his signature on c_h if and only if he has received the signatures of every other player on $\hat{c}_h$. In the enforcement stage $\hat{H}$, he complains to the center if and only if he has received all signatures on $\hat{c}_h$, but he has not received all signatures on c_h, or if some other player has complained.

The remainder of π^* for G and H is simple. If all signatures on c_h become known to all players, then the players play (a_i, a_{-i}) in G to achieve o, just as before, and then complain to the center in H if some player deviates. If, however, some player i deviates from the equilibrium in F (whether by choosing not to reveal in $\hat{F}$, choosing not to reveal in $\hat{G}$, or failing to complain in $\hat{H}$), in such a way that the signatures on c_h do not become commonly known, then the agents coordinate on that player's punishment equilibrium ρ^i. If several players deviate during F, the agents coordinate on the punishment equilibrium of the last player to deviate.

Our strategy π^* is now an equilibrium. Thus, we can achieve any outcome achievable with a busy center with a center that does no work in equilibrium.

Theorem 6 *Let X be the extended game $\hat{F} \cdot \hat{G} \cdot \hat{H} \cdot G \cdot H$, and let ρ^i be the punishment equilibrium for i in G. Then, for any $o_{(a_i,a_{-i})}$ such that for all i, $V_i(a_i, a_{-i}) \geq V_i(\rho^i)$, there exists a contract c_h for which there is an strategy pro le π^* such that π^* is a subgame perfect equilibrium of the extended game and $o_{(a_i,a_{-i})}$ is the equilibrium outcome of π^*. Furthermore, in π^*, the center takes no action during any stage.*

Proof: Let us sketch why this will be a subgame perfect equilibrium. We will proceed by backwards induction.

So long as no single player gains complete knowledge of the signatures on c_h, then π^* is a subgame perfect equilibrium in G and H. This does not occur if π^* is played in F, so it is suf cient to prove that π^* is a subgame perfect equilibrium in F. We will show that, even after one deviation, either all players know all the signatures on c_h or no player knows all signatures on c_h.

Consider the second round of $\hat{H}$. If no player complained in the rst round, second-round complaints have no effect. If a player complained in the rst round of $\hat{H}$, then it will be dominant for every other player to complain according to π^* to avoid the punishment of M from the center. Thus, all players will know all signatures on c_h.

Consider the rst round of $\hat{H}$. There are three cases to distinguish. First, if every player knows all signatures on both $\hat{c}_h$ and c_h, then complaining will have no effect. Second, if every player knows all signatures on $\hat{c}_h$, but only one player knows all signatures on c_h. Every other player will complain in the rs t round and i must therefore complain in the rst or second rounds to avoid losing M. Every player will learn all signatures. Third, if only one player i knows all the signatures on $\hat{c}_h$, then i will not know all the signatures on c_h. If i does not complain, no player will learn all signatures on c_h and players will coordinate on i's punishment equilibrium. If i does complain, all others will complain in the second round and all players will learn all signatures.

Consider the stage $\hat{G}$. There are now two cases to consider. First, suppose all players know all signatures on $\hat{c}_h$. Then no player i can bene t by failing to reveal his signature on c_h, since the other players will complain, i will complain to avoid punishment, and all players will end up learning all signatures on c_h. Second, suppose only one player i knows the signatures on $\hat{c}_h$ because he failed to reveal in $\hat{F}$. Then no other players will reveal their signatures, and, whether i reveals or not, all players will coordinate on i's punishment equilibrium.

Finally, consider the stage $\hat{F}$. Suppose one player i deviates in the stage $\hat{F}$ by failing to reveal his signature on the pre-contract $\hat{c}_h$. Then he alone will have all the signatures on $\hat{c}_h$, and no one else will reveal their signatures on c_h in $\hat{G}$. According to the equilibrium, i will complain to the center in stage $\hat{H}$, resulting in complete knowledge of c_h and the decision to play (a_i, a_{-i}). □

Conclusion

We have discussed the power of a helpful center in enabling a group of players to make contracts which require them to play a certain strategy or face penalties. Even if the center brings no money to the system and transfers money from the players only after receiving permission, the center is able to help the players achieve nearly any outcome of the game. Moreover, we nd that the center is still able to help the players achieve these outcomes in equilibrium, even if he does not monitor the game and does not participate on the equilibrium path - in other words, even when the center does no work in equilibrium beyond suggesting a contract.

In fact, if the contracts the center would suggest are common knowledge or determined by a negotiation stage between the agents, the center does no work whatsoever in equilibrium. Incidentally, we notice that the center makes a pro t to cover his costs whenever his services are used. These two properties are very important for a third party who wishes to in uence outcomes in strategic settings that occur frequently, such as in the online auction setting.

References

Garay, J. A., and MacKenzie, P. D. 1999. Abuse-free multi-party contract signing. In *International Symposium on Distributed Computing*, 151–165.

Hafner, K. 2004. Vigilantes attack eBay fraud. *The New York Times*. Available at http://news.com.com/2100-1038_3-5176525.html.

Mas-Colell, A.; Whinston, M.; and Green, J. 1995. *"Microeconomic Theory"*. Oxford University Press.

Monderer, D., and Tennenholtz, M. 2003. k-implementation. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, 19–28.

Osborne, M., and Rubinstein, A. 1994. *A Course in Game Theory*. MIT Press.

Shoham, Y., and Tennenholtz, M. 1997. On the emergence of social conventions: Modeling, analysis, and simulations. *Arti cial Intelligence* 94:139–166.

Computing Best-Response Strategies via Sampling

Rob Powers
powers@cs.stanford.edu
Computer Science Department
Stanford University
Stanford, CA 94305

Yoav Shoham
shoham@cs.stanford.edu
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

We look at the problem of computing a best-response to an opponent's strategy, when this strategy is not known exactly but can instead be sampled. We rst give analytic results on the number of samples required in order to approximate the optimal best response. We then show experimentally, on a variety of games, that one can closely approximate the best response with a much smaller number of samples than are required by the formal guarantees. Finally, we go beyond so-called *oblivious sampling*; that is, we consider what happens if the opponent is aware that the agent has taken the samples, if the agent knows that the opponent is aware, and so on to higher levels of mutual modelling.

1 Introduction

Recent years have seen much research in AI on game theoretic multi-agent systems. Since most of game theory abstracts away from computational issues, much of the recent activity has concentrated on addressing algorithmic considerations. In this work we consider a particular algorithmic issue, namely computing a best response for one agent to a x ed opponent (when the strategies are x ed, there is no greater generality in considering a set of opponents, since they can be viewed as a single player with an expanded action space). We use the terms 'agent' and 'opponent' to indicate the different status of the two players in the problem we consider, but we interpret the term 'opponent' neutrally (in particular, it does not mean that the game is necessarily adversarial).

Although there are many other outstanding computational issues in game theory (notably, computing one or all Nash equilibria (Papadimitriou 2001)), best-response computation is arguably the most relevant to AI. There are several variants of the problem; for example, computing a best response against a known opponent or a distribution of opponents. The latter, for example, is the natural computational question in the context of competitions such as the Trading Agent Competition (Wellman & Wurman 1999). Even against a known opponent, however, the problem is in general intractable. (see Gilboa & Zemel 1989) The problem of computing a best response to a bounded automata was shown to be NP-complete in (Papadimitriou 1992), while nding the best response for an arbitrary strategy can be non-computable in some instances (Nachbar & Zame 1996).

Besides these complexity results, there are other reasons why one might not be able to engage in a straightforward best-response computation. For one thing, the opponent's strategy might be too complicated to represent. Worse, it might not be available at all. However, recalling that the opponent's strategy is a distribution over its pure strategies, we can sometimes sample from this distribution. There are many real-world examples where such a situation could arise naturally. For instance, a government might capture some members of a terrorist organization and learn their strategies. This sample can then be used to estimate the strategies of the remaining agents. In a similar example one could study the code/strategy of a set of known computer viruses and use this information to design effective security against new viruses. Or in a corporate setting, companies regularly hire away members of their competitors organization, giving them access to those employees knowledge/strategies. For organized games such as chess or poker, there are large reservoirs of data on samples both from records of previous games played and books written describing proposed strategies for play.

We propose to analyze these situations, where the agent has access to some number of samples from its opponent's distribution and uses them to calculate a best response. Several questions suggest themselves, including the following:

- Theoretically speaking, how many samples do we need to approximate the best response within a given constant?

- Empirically speaking, how good is a best response that is based on a small sample?
- What happens if the opponent is aware that the agent has the samples, if the agent is aware of this, and so on as one increases the levels of outguessing?

In the next section, we show a bound on the number of samples required to guarantee a payoff within ϵ of the true best response. The following section looks at the empirical question, analyzing payoff achieved as a function of samples across different game environments. Both of these sections assume the opponent is oblivious to the fact that the agent is taking samples. To address this limitation, we then consider the results of mutual modelling, when the opponent is aware that the samples have been taken. Finally we conclude with a few general observations and areas for future work.

2 Oblivious Sampling - Theoretical Bounds

Given our approach of drawing samples from an opponent's distribution over strategies, a natural question is how many samples are required to achieve a certain level of performance. Using the Hoeffding inequality we can nd the minimum number of samples required to guarantee with probability $1 - \delta$ that the best response we compute achieves within ϵ of the payoff of the best response to the true distribution (Hoeffding 1956). Let us assume that each of our samples is an independent and identically distributed random variable drawn from the true distribution and that all payoffs are bounded within [0,1]. When computing the best response we are effectively calculating the payoff of each of the agent's k strategies from the set A against the distribution of the m samples in S and choosing the action with the highest payoff. Let

$$\hat{a} = \arg\max_{a \in A} \hat{V}(a), \tag{1}$$

where

$$\hat{V}(a) = \frac{1}{m} \sum_{s \in S} R(a, s) \tag{2}$$

and $R(a, o)$ is the agent's payoff function for the game when the agent plays a and the opponent plays o. Letting a^* be the true best response, we want

$$V(\hat{a}) \geq V(a^*) - \epsilon, \tag{3}$$

where $V(a)$ is the value achieved by strategy a against the true distribution. Since $\hat{V}$ is the mean of m iid random variables, we know by the Hoeffding inequality (also known as the Chernoff bound) that for any a,

$$P(|\hat{V}(a) - V(a)| > \gamma) < 2e^{-2\gamma^2 m}. \tag{4}$$

By the union bound in probability theory, we know that

$$P(\exists a : |\hat{V}(a) - V(a)| > \gamma) \leq \sum_{a \in A} P(|\hat{V}(a) - V(a)| > \gamma)$$

$$\leq k * 2e^{-2\gamma^2 m}$$

$$P(\forall a |\hat{V}(a) - V(a)| < \gamma) \geq 1 - 2ke^{-2\gamma^2 m}. \tag{7}$$

In particular if we have

$$|\hat{V}(\hat{a}) - V(\hat{a})| \leq \gamma, \tag{8}$$

$$|\hat{V}(a^*) - V(a^*)| \leq \gamma \tag{9}$$

and by (1)

$$\hat{V}(\hat{a}) \geq \hat{V}(a^*), \tag{10}$$

we get

$$V(\hat{a}) \geq V(a^*) - 2\gamma. \tag{11}$$

Setting $\gamma = \frac{\epsilon}{2}$ and $2ke^{-2\gamma^2 m} = \delta$, we can solve for m to get a bound of

$$m \geq \frac{2}{\epsilon^2} \log(\frac{2k}{\delta}) \tag{12}$$

which will guarantee that the best response to the samples is within ϵ of the actual best response with probability $1-\delta$. Note that this bound is independent of the number of possible opponent strategies and that it depends only on the log of the size of the agent's strategy space. This lets us bypass the potential complexity of the opponent's true strategy distribution by evaluating only the set of agent strategies required to include a best response to any possible distribution. For instance, although the set of possible mixed strategies is in nite, the set of pure strategies is guaranteed to contain a best response to any mixed strategy. Although the existence of these bounds is encouraging, the Hoeffding inequality is well-known to be quite weak by the learning theory community. Further work has focused on tightening the bounds, reducing the dependence on ϵ from $\frac{1}{\epsilon^2}$ to $\frac{1}{\epsilon}$ in many situations. The question remains, however, how many samples would be required in practice, which will be the focus of the following section.

3 Oblivious Sampling - Empirical Results

While the theoretical results are encouraging, the number of samples required could still prove impractical in many instances. In order to test the empirical performance of sampling, we performed a series of tests for different types of arti cial games and distributions of opponent strategies. We found that the sampling performed quite well in all of the domains and often achieved close to optimal performance with only a small number of samples. We initially divided the domains on the basis of their payoff functions, analyzing zero-sum, common-payoff, and general-sum games independently. However, the performance we observed seemed independent of these categories. On re- ection, this is not particularly surprising for a x ed opponent distribution since it is oblivious to the samples being taken. Given this situation, only the agent's payoffs are relevant in determining a best response.

In order to illustrate these ndings, we will show results from two individual games: one-card poker and a repeated game of chicken, as well as aggregate results for randomly generated games. For each setting, an opponent distribution is rst selected and then m samples are drawn from this distribution and provided to the agent. The agent calculates a best response to this set of samples, weighting all samples equally, and this best response strategy is tested empirically against opponent strategies drawn from the true distribution.

In our rst experiment, we de ned a game based on a simpli ed version of poker. At the start of the game each of two players receives a card uniformly selected from the set of integers $[1...N]$. Each player then antes 1 chip, and a player is randomly chosen to start betting. On their turn, each player can choose either to bet B chips or pass, play then continues to the other player until either both players have passed or one player folds (passes when the opponent has bet a larger amount). If one player folds the other player wins the contents of the pot, otherwise the player with the higher card wins (the pot is split in the case of a tie). The results shown here are for the game where each player is limited to a single bet. This gives each player a total of 12 distinct betting strategies which they can condition on the card they receive, resulting in 12^N possible pure strategies.

We ran tests against three different opponent distributions. The rst was a uniform distribution assigning equal probability to playing any of the 12^N pure strategies. The next assigned a Gaussian distribution in strategy space about a random pure strategy, while the nal used a probability distribution that combined two such Gaussians. The results are shown in Figure 1 for games with bets of 4 units and a deck of 10 cards. Similar results were also obtained for games with a variety of values for B and N. Although not shown, even in this zero-sum game the mini-max strategy fares worse than even a small number of samples, since it fails to take advantage of the opponent's liabilities.

For our next setting, we considered a repeated version of the game of Chicken. Each player has a choice of two actions at each time step, either to 'Dare' or 'Yield'. If both players 'Dare', the result is the worst possible outcome for each player, but a player can achieve its maximum reward by daring while its opponent yields. In Figure 2 we show the payoffs for the three versions of the game of chicken we used in our experiments. In the repeated setting, the strategy space allows the players to base their current actions on the history of the last H outcomes within the game. Note that we restrict the agent as well to only using information from the last H outcomes during the repeated game, thus resulting in essentially a one-shot game. Since there are 4 possible outcomes, this results in a strategy space of size 2^{4^H}. Once again we obtained quick convergence to the optimal response value with a relatively small number of samples, despite the huge strategy space. The results are qualitatively quite similar to those for single-card poker, so we have omitted the graph, although this setting will be referred to again in the following section.

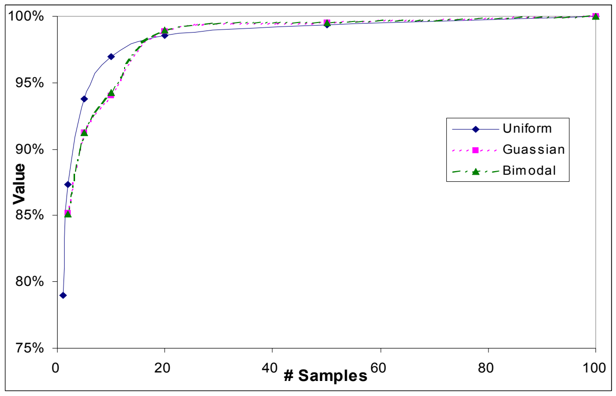


Figure 1: Percent of maximum payoff achieved using the given number of samples in single-card poker.

Finally, experiments were performed for randomly generated matrix games. Each game has k actions for each agent with the payoffs randomly distributed in the range $[0, 1]$. For these games we added an additional opponent distribution which was forced to assign zero probability to strategies employing dominated actions. Results for general-sum games of different sizes against this new opponent distribution are shown in Figure 3. Results are basically identical against a uniform distribution over opponents or when the payoffs are restricted to be common-payoff. While the agent can still achieve close to optimal payoffs, the number of samples required increases more rapidly with the number of actions than in the previous games, possibly indicating an advantage of the structure in the previous games when using sampling.

From the empirical results we can see that sampling is effective in approximating a best response across a wide variety of different environments. Most of these environments allow us to achieve good performance with far fewer samples then our theoretical bounds require. We can de ne a

	$Dare$	$Yield$
$Dare$	$0, 0$	$4, 1$
$Yield$	$1, 4$	$2, 2$

	D	Y
D	$0, 0$	$3, 1$
Y	$1, 3$	$2, 2$

	D	Y
D	$0, 0$	$8, 5$
Y	$5, 8$	$6, 6$

Figure 2: Single round payoffs in three games of Chicken.

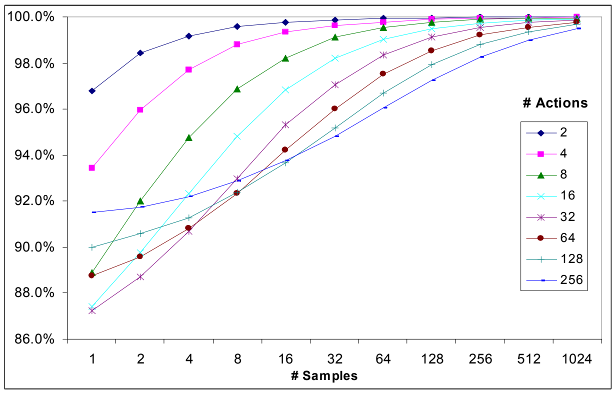


Figure 3: Percent of maximum payoff achieved for random matrix games of different sizes.

metric for the ease with which this is possible by setting a performance guarantee we wish (such as 99% of optimal) and then comparing the sample complexity for achieving this performance across games. Figure 4 shows the values for such a metric applied to the games analyzed here.

4 Beyond Oblivious Sampling

So far, we've assumed that the agent is the only one allowed to explicitly reason about their opponent. They are allowed to gather samples from the opponent's distribution and then compete against the original distribution unaffected by their actions. What if the opponent is aware that the samples have been taken? Presumably they might use this information to change their strategy in response to the new situation. We can then ask a number of questions about this setting.

- How should the opponent use its awareness that samples were taken and how should the agent respond in turn?
- When and to whom is it an advantage to explicitly model the other player?

Environment	90%	95%	99%
Poker (N=10, B=4, Uniform Dist.)	5	10	20
Poker (N=10, B=4, Gaussian Dist.)	5	15	25
Poker (N=10, B=4, Bimodal Dist.)	5	15	25
Chicken (Payoffs 4_2_1, Uniform)	1	8	100
Chicken (Payoffs 8_6_5, Uniform)	3	8	20
Random Games (4 Actions)	1	2	10
Random Games (16 Actions)	3	10	64
Random Games (64 Actions)	3	25	250
Random Games (256 Actions)	1	50	500

Figure 4: Samples required to approximate best response.

- What happens in the limit of in nite mutual modelling?

First we need to consider what kind of information the opponent might have. They could know only that some samples were taken, and nothing about the value or number of samples. In this case it seems the only possible improvement the opponent could make in their strategy would be to assume the agent has calculated a best response to their true distribution and play a best response to that. However, if the opponent in addition knows how many samples were taken (or has a distribution over likely sample sizes) they can improve their performance. This may seem surprising, but consider the game shown in Figure 5, with the opponent's original distribution consisting of 75% agents which always play L and 25% agents which always play R.

If the agent calculates the true best response, it will always play B, forcing the opponent (the column player) to always play L. However, if the agent only receives one sample, then 75% of the time it will choose T as its best response instead. Now, playing L only yields a payoff of 1, while playing R yields an expected payoff of 3 $(75\% \times (V(T, R) = 4) + 25\% \times (V(B, R) = 0))$. In this example, if the agent has taken many samples their play will likely correspond to the best response strategy for the true distribution, however, there is no guarantee this is true in general. While we showed earlier that the value achieved by the agent will approach the value of the best response, this does not in general imply that the play will approach the best response strategy for any nite number of samples. The agent may have a strategy that yields within ϵ of the maximum value but lies arbitrarily far away in the strategy space. In this case, there is no guarantee that playing the best response to the true best response is a good strategy against the ϵ-best response actually played. In practice this tends to work well for large numbers of samples, but is dominated by employing the correct Bayesian calculation using the actual number of samples taken. The exact calculation can prove infeasible for large strategy spaces, but once again we can employ sampling to approximate it, this time from the opponent's point of view. The opponent can randomly draw sets of samples from its original true distribution to match the samples the agent may have taken and use the best responses to these samples sets as its samples from the agent's true distribution. In Figure 6 we can see the results of using this method for a variety of sample sizes. The measure here is how much value is lost by the

$$\begin{array}{c|c|c|}
 & L & R \\
\hline
T & 2,1 & 1,4 \\
\hline
B & 1,1 & 5,0 \\
\hline
\end{array}$$

Figure 5: Game showing the value of knowing the sample size.

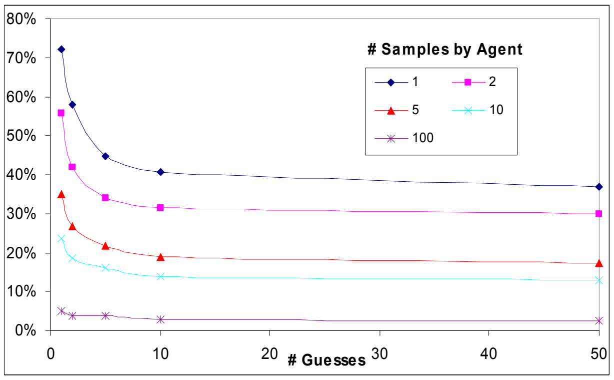


Figure 6: Opponent's loss of value in simple poker setting versus that achieved with full information on samples taken for different agent sample sizes.

opponent relative to the value they could have achieved if they knew the exact samples taken by the agent. We can see that this improves as the opponent simulates more sample sets and that the loss from not knowing the samples is highest when a single sample was taken.

Next, we consider the case in which the opponent knows not only the number of samples but also the value of each sample. They could assume the agent would calculate a best response to those samples and therefore calculate their own best response to the agent's strategy. Of course, this process can escalate inde nitely , as the agent realizes the opponent knows the samples were taken, etc.

For the settings we described previously, we analyzed the results of this recursive modelling under different assumptions about the level of mutual modelling carried out by each agent. Results for the individual games are discussed in the following subsections, organized into zero-sum, common-payoff, and general-sum games.

Note that for these results we assumed that at least one player had correct beliefs about the other player and also that the other player's beliefs were accurate except for not accounting for the last level of analysis performed. As the players beliefs depart further from the actual situation, the payoffs tend to move towards those achieved by a random strategy. This generally corresponds to a decrease in value for each agent for team games and many general sum games.

4.1 Zero-sum Games

Within the simple poker game, as each player performs additional introspection about the other's strategy, the magnitude of the payoffs increases until the play settles into a simple cycle of alternating strategies. In this cycle, the primary driver of value is holding the correct beliefs about the level of analysis the other player has performed. When

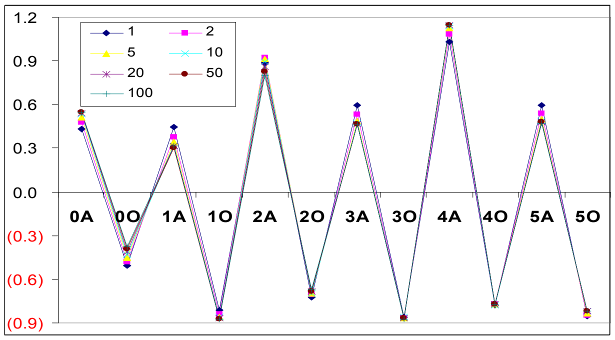


Figure 7: Payoffs for increased levels of introspection

holding correct beliefs each player is guaranteed at least the value of the game, since they are performing a best response calculation. Payoffs for the sampling agent in the simpli ed poker game are shown in Figure 7. The points labelled 0A correspond to the agent calculating a best response to the samples received without the opponent changing strategies. Each point to the right then advances the level of mutual modelling by one (with the letter indicating which player has the informational advantage). The number of samples taken has little impact on payoff and mainly serves to decrease the magnitude of the oscillations as the number of samples increases. Presumably this is because the strategies found with larger numbers of samples are slightly more robust.

4.2 Common-payoff Games

In our empirical experiments, the play would often converge rapidly to the Pareto optimal Nash equilibrium of the game as the players reasoned more deeply about one another. In general, the players are guaranteed to converge to playing a Nash equilibrium for some nite amount of mutual modelling (assuming a nite action space and generic payoffs). This follows since each player has full knowledge of the best response calculated by the other for any level of reasoning. Given this, each subsequent best response calculation is guaranteed to be monotonically nondecreasing in the common value achieved by all players. For two-player games or games with generic payoffs, the players will not enter a cycle where they return to a previously encountered outcome until they reach a x ed-point outcome. Since there are a nite number of distinct outcomes, they must eventually arrive at an outcome where each player is playing its best response, meeting the de nition of a Nash equilibrium.

4.3 General-sum Games

We ran additional tests on the game of centipede. In this game, each player alternates in choosing whether to stop the game or continue for up to one hundred turns. Each

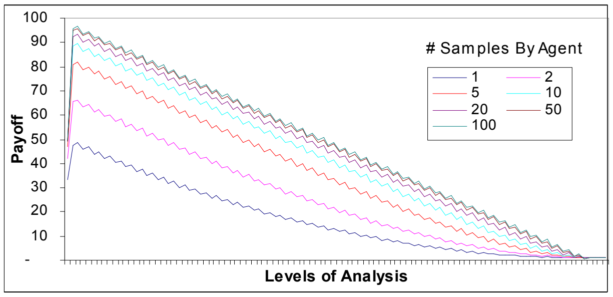


Figure 8: Value achieved by each player for successive levels of mutual modelling in the centipede game.

player then receives a reward equal to the number of turns the game continued plus a bonus reward of two for the player who chose to stop the game. Looking at Figure 8, we can see that there is at rst a jump in value to both players since the agent's initial best response is near the strategy of always continuing. The value then drops steadily for increased introspection, matching the process of backwards induction. (If no player would ever continue past k steps, then the best response is to always stop the game at k-1 steps.) The limit of this process is the unique Nash equilibrium, where each player chooses to stop immediately.

In the repeated game of chicken, play once again converges to a Nash equilibrium in the limit. However, this game has multiple Nash equilibria corresponding to different players choosing 'Dare' when their opponent chooses 'Yield'. The beliefs in the game will tend to converge to the Nash equilibrium closest in the strategy space to the agent's initial best response, which is a function of the particular payoffs. In Figure 9, we can see the payoffs to each player for increasing amounts of mutual modelling by both players. Again, each point to the right along the x-axis indicates one more level of analysis by one of the players.

Looking at these two games we can address the question of whether informational advantage always offers a better payoff. It turns out that the answer depends on who cur-

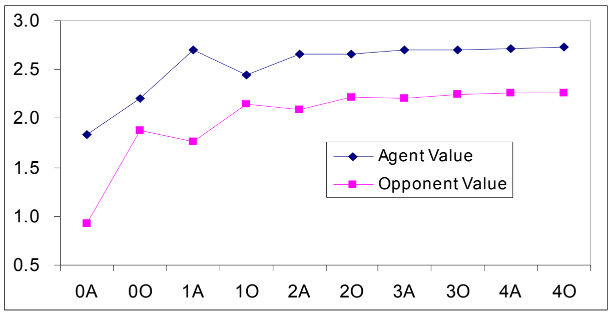


Figure 9: Value achieved by each player for successive levels of mutual modelling in a repeated game of chicken.

	H	T	X
H	$1,-1$	$-1,1$	$-2,-2$
T	$-1,1$	$1,-1$	$-2,-2$
X	$-2,-2$	$-2,-2$	$1,1$

Figure 10: Sample game in which recursive modelling does not converge to a Nash equilibrium.

rently has the advantage. Unsurprisingly, if the opponent currently has the advantage, it can never hurt the agent's payoff (in the short turn) to do further analysis to take the advantage, since this is a best-response calculation. However, if the agent currently has the advantage, the opponent taking the advantage could help. This is particularly true in team games where all players are maximizing the same function, but can also occur in some general-sum games, such as the repeated game of chicken shown above. Notice in Figure 9, that when the opponent rst starts to reason about the agent taking samples that the payoffs to both players go up.

Given that these games have all converged to Nash equilibria, will this process in general converge to a pure strategy Nash equilibrium when one exists? If the process does converge, it must be to a Nash equilibrium since each player calculates best response. However, it turns out not to converge in all games since it is possible for the best responses to cycle through outcomes without encountering an equilibrium. Consider the game shown in Figure 10 where the agent received a sample opponent strategy that always plays T. One can see that mutual modelling will result in each player alternating between always playing H or always playing T without ever reaching either the mixed Nash of playing H and T each with 50% probability or the pure Nash equilibrium with both players playing X.

5 Conclusions and Future Work

Most of our results fall in line with our predictions stated in the introduction, although not without a few surprises.

- Sampling can be shown to approximate the true best response with a number of samples logarithmic in the size of the agent's strategy space.
- Empirical results show sampling to be effective against a variety of hidden opponent distributions and different game environments.
- Analyzing successive mutual modelling by the players results in either convergence to equilibrium or cyclic behavior, reminiscent of ctitious play in repeated games. (Brown 1951)

- Even a small difference in player knowledge, such as information about the number of samples taken, can produce a major difference in the value an agent can achieve.
- While in general the advantage from modelling falls to the player with correct beliefs, games with some degree of cooperation allowed one player to bene t from increased mutual modelling by the other player.

Several of our ndings encourage further study of this approach. For many of the settings tested, the empirical performance far exceeded the theoretical guarantees for small numbers of samples. Although the bounds were known to be weak, can we more clearly characterize the factors leading to settings where sampling is most effective? One possible factor relates to qualities of the agent's strategy space. In the proof we assumed a worst case distribution of payoffs for the agent's strategies. In practice the effective space of strategies that need be considered may be much smaller for many games. For instance, there may be many strategies that achieve (nearly) optimal payoffs against the true distribution, creating a wider target to choose from. Also, there may be many strategies that fail to be best responses against (almost) any strategy selected by the samples. These strategies can then be ignored in the calculation since there is no chance of their being confused with an optimal strategy. A rst step in better de ning the space of games where sampling is effective could be achieved by testing the approach over a wide sample of games and opponent distributions. A machine learning approach could then be used to learn what parameters affect the number of samples required. Given this understanding, it may be possible to tighten the theoretical bounds on the number of samples required by adding additional restrictions on the strategy space and the payoffs of the game.

While both agents could pro t in general-sum games with some coordination, the exact values achieved could vary signi cantly based on the starting samples, as seen in the Repeated Chicken game. This opens up a further question of how an opponent might take advantage of this dependence. What if the opponent had the ability to manipulate the samples the agent received? This could occur either through adding additional samples of their choosing (planted evidence or double agents) or changing the distribution the agent is sampling from.

Additionally, all of our detailed analysis on recursive modelling assumed that the agents had certainty over the information they possessed about the other player (even when one agent was wrong). What if we instead allow the agents to have more complex beliefs over the possible information or beliefs of the other agent? This may allow the agents to develop more robust strategies as a result of their uncertainty. Note that as long as the agents can characterize their uncertainty precisely they can form a probability distribution over possible opponent strategies and still calculate a pure strategy best response. If however they believe their opponent may outguess them in unforseen ways, there can be pressure to play mixed strategies that reduce the penalty of losing the informational advantage. Given that equilibria are often justi ed as the endpoints of in nite analysis among perfectly rational opponents, are there any reasonable constraints we could add to the player's process of mutual modelling such that in the limit it would also converge to an equilibrium?

In terms of direction for our future work, probably the greatest limitation of the current approach is that it requires that the samples from the opponent's distribution be entire strategies. For instance, in the simple poker setting each sample contains the betting strategy for the opponent given any possible card. In many situations it is much easier to acquire part of a strategy, such as the on-path play for a previous game. The question of how best to utilize this weaker information would help expand the applicability of this work to a wider class of settings.

Acknowledgements

This work was supported in part by DARPA grant F30602-00-2-0598, NSF grant IIS-0205633, and a Benchmark Stanford Graduate Fellowship. The authors would like to thank Andrew Ng for his invaluable comments and assistance in calculating the error bounds for oblivious sampling and also the members of the Stanford University Multi-Agent Research Group for their feedback.

References

G. Brown (1951). Iterative Solution of Games by Fictitious Play. In *Activity Analysis of Production and Allocation*. New York: John Wiley and Sons.

I. Gilboa and E. Zemel (1989). Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior* 1:80-93.

W. Hoeffding (1956). On the distribution of the number of successes in independent trials. *Annals of Mathematical Statistics* 27:713-721.

J. Nachbar and W. Zame (1996). Non-computable strategies and discounted repeated games. *Economic Theory* 8:103-122.

C. Papadimitriou (1992). On players with bounded number of states. *Games and Economic Behavior* 4:122-131.

C. Papadimitriou (2001). Algorithms, Games, and the Internet. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computer*, 749-753.

M. Wellman and P. Wurman (1999). A trading agent competition for the research community. In *IJCAI-99 Workshop on Agent-Mediated Electronic Trading*.

Dispersion Games: General Definitions and Some Specific Learning Results*

Trond Grenager and **Rob Powers** and **Yoav Shoham**
Computer Science Department
Stanford University
Stanford, CA 94305
{grenager, powers, shoham}@cs.stanford.edu

Abstract

Dispersion games are the generalization of the *anti-coordination game* to arbitrary numbers of agents and actions. In these games agents prefer outcomes in which the agents are maximally dispersed over the set of possible actions. This class of games models a large number of natural problems, including load balancing in computer science, niche selection in economics, and division of roles within a team in robotics. Our work consists of two main contributions. First, we formally define and characterize some interesting classes of dispersion games. Second, we present several learning strategies that agents can use in these games, including traditional learning rules from game theory and artificial intelligence, as well as some special purpose strategies. We then evaluate analytically and empirically the performance of each of these strategies.

Introduction

A natural and much studied class of games is the set of so-called *coordination games*, one-shot games in which both agents win positive payoffs only when they choose the same action (Schelling 1960).[1] A complementary class that has received relatively little attention is the set of games in which agents win positive payoffs only when they choose distinct actions; these games have sometimes been called the *anti-coordination games*. Most discussion of these games has focused only on the two-agent case (see Figure 1), where the coordination game and the anti-coordination game differ by only the renaming of one player's actions. However, with arbitrary numbers of agents and actions, the two games diverge; while the generalization of the coordination game is quite straightforward, that of the anti-coordination game is more complex. In this paper we study the latter, which we call *dispersion games* (DGs), since these are games in which agents prefer to be more dispersed over actions.[2] Although one can transform a dispersion game into a coordination game in which agents coordinate on a maximally dispersed assignment of actions to agents, the number of such assignments grows exponentially with the number of agents.

	A	B
A	1	0
B	0	1

	A	B
A	0	1
B	1	0

Figure 1: Two-agent coordination game (left) and anti-coordination game (right).

DGs model natural problems in a number of different domains. Perhaps the most natural application is presented by the much studied *load balancing* problem (see, e.g., Azar *et al.* 2000). This problem can be modeled as a DG in which the agents are the users, the possible actions are the resources, and the equilibria of the game are the outcomes in which agents are maximally dispersed. Another natural application of DGs is presented by the *niche selection* problem studied in economics and evolutionary biology. In a general niche selection problem, each of n oligopoly producers wishes to occupy one of k different market niches, and producers wish to occupy niches with fewer competitors. Other niche selection problems include the *Santa Fe bar problem* proposed by Arthur (1994), and the class of *minority games* (Challet & Zhang 1997). These niche selection problems can all be modeled in a straightforward manner by DGs. Finally, we note that DGs can also serve as a model of the process of role formation within teams of robots. In fact, the initial motivation for this research came from empirical work on reinforcement learning in RoboCup (Balch 1998).

This paper makes two types of contributions. First, we formally define and characterize some classes of DGs that possess special and interesting properties. Second, we analyze and experimentally evaluate the performance of different learning strategies in these classes of games, including two standard learning rules from game theory and artificial intelligence, as well as two novel strategies. The remainder of this article is organized as follows. In the first section we present the game definitions. In the second and third sections we present the learning strategies and the results and analysis of their performance. Finally, in the last section we

*This work is supported in part by DARPA Grant F30602-00-2-0598 and by a Benchmark Stanford Graduate Fellowship.

[1]In this paper, we assume familiarity with basic game theory; our formulations are in the style of (Osborne & Rubinstein 1994).

[2]We chose this name after (Alpern 2001) who studies a subclass of these games which he calls *spatial dispersion problems*.

discuss these findings and present ideas for future research.

Dispersion Game Definitions

In this section we begin by discussing some simple dispersion games, and work our way gradually to the most general definitions. All of the DGs we define in this section are subclasses of the set of *normal form games*, which we define as follows.

Definition 1 (CA, CP, CACP games) *A* normal form game *G is a tuple $\langle N, (A_i)_{i \in N}, (\succeq_i)_{i \in N} \rangle$, where*

- *N is a finite set of n agents,*
- *A_i is a finite set of actions available to agent $i \in N$, and*
- *$\succeq_i$ is the preference relation of agent $i \in N$, defined on the set of outcomes $O = A^n$, that satisfies the von Neumann-Morgenstern axioms.*

A game G is a common action (CA) game if there exists a set of actions A such that for all $i \in N$, $A_i = A$; we represent a CA game as $\langle N, A, (\succeq_i)_{i \in N} \rangle$. Similarly, a game is a common preference (CP) game if there exists a relation $\succeq$ such that for all $i \in N$, $\succeq_i = \succeq$; we represent a CP game as $\langle N, (A_i)_{i \in N}, \succeq \rangle$. We denote a game that is both CA and CP as CACP. We represent a CACP game as $\langle N, A, \succeq \rangle$

Note that we use the notation $\langle a_1, \dots, a_n \rangle$ to denote the outcome in which agent 1 chooses action a_1, agent 2 chooses action a_2, and so on. In a CA game where $|A| = k$, there are k^n total outcomes.

Common Preference Dispersion Games

Perhaps the simplest DG is that in which n agents independently and simultaneously choose from among n actions, and the agents prefer only the outcomes in which they all choose distinct actions. (This game was defined independently in (Alpern 2001).) We call these outcomes the *maximal dispersion outcomes* (MDOs).

This simple DG is highly constrained. It assumes that the number of agents n is equal to the number of actions k available to each agent. However, there are many problems in which $k \neq n$ that we may wish to model with DGs. When $k > n$ the game is similar to the $k = n$ game but easier: there is a larger proportion of MDOs. When $k < n$ however, the situation is more complex: there are no outcomes in which all agents choose distinct actions. For this reason, we will need a more general definition of an MDO. In the definitions that follow, we use the notation n_a^o to be the number of agents selecting action a in outcome o.

Definition 2 (MDO) *Given a CA game G, an outcome $o = \langle a_1, \dots, a_i, \dots, a_n \rangle$ of G is a* maximal dispersion outcome *iff for all agents $i \in N$ and for all outcomes $o' = \langle a_1, \dots, a_i', \dots, a_n \rangle$ such that $a_i' \neq a_i$, it is the case that $n_{a_i}^o \leq n_{a_i'}^{o'}$.*

In other words, an MDO is an outcome in which no agent can move to an action with fewer other agents. Note that when the number of agents is less than or equal to the number of actions, an MDO allocates exactly one agent to each action, as above.

Under this definition, the number of MDOs in a general CA game with k actions is given by

$$MDO(n,k) = n! \frac{\binom{k}{n \bmod k}}{\lceil n/k \rceil^{n \bmod k} \lfloor n/k \rfloor !^k}.$$

When $k = n$ this expression simplifies to $n!$, since there are $n!$ ways to allocate n agents to n actions.

The simple DG presented above also makes another strong assumption. It assumes that an agent's preference over outcomes depends only on the overall configuration of agents and actions in the outcome (such as the number of agents that choose distinct actions), but not on the particular identities of the agents or actions (such as the identities of the actions that are chosen). We call these the assumptions of *agent symmetry* and *action symmetry*. However, many situations we might like to model are not agent and action symmetric. For example, role formation on soccer teams is not action symmetric. The identity of a particular field position in an outcome can affect the performance of the team: a team with a goalie but no halfback would probably perform better than one with a halfback but no goalie, all else being equal. Robot soccer is also not necessarily agent symmetric. If agent 1 is a better offensive than defensive player, then a team may perform better if agent 1 is a forward instead of a fullback, all else being equal. We use the following formal definitions of symmetry.

Definition 3 (Agent Symmetry) *A CA game $G = \langle N, A, (\succeq_i)_{i \in N} \rangle$ is* agent symmetric *iff for all outcomes $o = \langle a_1, \dots, a_i, \dots, a_n \rangle$, and for all permutations $o' = \langle a_1', \dots, a_i', \dots, a_n' \rangle$ of o, for all $i \in N$, $o \succeq_i o'$ and $o' \succeq_i o$.*

Definition 4 (Action Symmetry) *A CA game $G = \langle N, A, (\succeq_i)_{i \in N} \rangle$ is* action symmetric *iff for all outcomes $o = \langle a_1, \dots, a_i, \dots, a_n \rangle$ and $o' = \langle a_1', \dots, a_i', \dots, a_n' \rangle$, if there exists a one-to-one mapping $f : A \to A$ such that for all $i \in N$, $f(a_i) = a_i'$, then for all $i \in N$, $o \succeq_i o'$ and $o' \succeq_i o$.*

In fully symmetric games, agents cannot distinguish between outcomes with the same configuration of numbers of agents choosing actions. Thus we use the abbreviated notation $\{n_1, \dots, n_k\}$ to refer to the set of outcomes in which n_1 agents choose some action, n_2 agents choose a different action, and so on. By convention, we order the actions from most to least populated.

We are now ready to state the formal definition of a weak DG that is well defined over the set of all CACP games, including asymmetric games and games with arbitrary n, k.

Definition 5 (Weak DG) *A CACP game $G = \langle N, A, \succeq \rangle$ is a* weak dispersion game *iff the set of $\succeq$-maximal outcomes of G is a subset of the set of MDOs of G.*

This definition requires only that at least one of the MDOs is a preferred outcome, and that none of the non-MDOs is a preferred outcome. This definition is weak because it places no constraints on the preference ordering for the non-maximally-preferred outcomes.[3] For this reason, we also

[3]The reader may wonder why our definitions don't require that

state a strong definition. Before we can state the definition, however, we will need the following *dispersion relation*.

Definition 6 ($\sqsupseteq$) *Given two outcomes $o = \langle a_1, \ldots, a_i, \ldots, a_n \rangle$ and $o' = \langle a'_1, \ldots, a'_i, \ldots, a'_n \rangle$, we have that $o \mathbf{D} o'$ iff there exists a agent $i \in N$ such that $a'_i \neq a_i$, and $n^o_{a_i} < n^{o'}_{a'_i}$, and for all other agents $j \in N, j \neq i$, $a_j = a'_j$. We let the* dispersion relation $\sqsupseteq$ *be the reflexive and transitive closure of* $\mathbf{D}$.

In other words, o is more dispersed than o' if it is possible to transform o' into o by a sequence of steps, each of which is a change of action by exactly one agent to an action with fewer other agents. It is important to note that the dispersion ordering is a structural property of any CACP game. The dispersion relation over the set of outcomes forms a partially ordered set (poset). Note that the set of MDOs is just the set of $\sqsupseteq$-maximal elements of O.

There are many other measures that we could use instead of the qualitative dispersion relation. Entropy is consistent with, but stronger than our dispersion relation: if $o \sqsupseteq o'$ then the entropy of o is higher than that of o', but the converse is not necessarily true. We have chosen to base our definitions on the weaker dispersion relation because it is the most general, and because it corresponds directly to a single agent's change of actions.

Using this dispersion relation, we can state the formal definition of strong DGs.

Definition 7 (Strong DG) *A CACP game $G = \langle N, A, \succeq \rangle$ is a* strong dispersion game *iff for all outcomes $o, o' \in O$, it is the case that if $o \sqsupseteq o'$ but not $o' \sqsupseteq o$, then $o \succeq o'$ but not $o' \succeq o$.*

Recall that the preference relation $\succeq$ forms a total ordering while the dispersion relation $\sqsupseteq$ forms a partial ordering. Thus this definition requires that o is strictly preferred to o' when o is strictly more dispersed than o'.

If the strong definition has such nice properties, why bother to state the weak definition at all? There are many situations which have a dispersion quality but which cannot be modeled by games in the stronger class. Consider the situation faced by Alice, Bob, and Charlie who are each choosing among three possible roles in the founding of a company: CEO, COO, and CFO. Because they will be compensated as a group, the situation can be modeled as a CP game. However, suppose that Bob would be a terrible CEO. Clearly, the agents would most prefer an outcome in which each role is filled and Bob is not CEO; thus the game satisfies the weak definition. However, rather than have all roles filled and Bob alone be CEO, they would prefer an outcome in which Bob shares the CEO position with one of the other agents (i.e., both Bob and another agent select the "CEO" action), even though it leaves one of the other roles empty. In other words, the preference relation conflicts with the dispersion ordering, and the game does not satisfy the strong definition.

all MDOs are maximal outcomes. In fact, it is easy to verify that this must be the case in a fully symmetric DG.

Non-Common-Preference Dispersion Games

There are also several interesting classes of non-CP dispersion games we might like to model. Due to space considerations we will not define these classes formally, but instead present a few motivating examples.

Consider again the load balancing application in which each of n users simultaneously wishes to use one of k different resources. If the users all belong to a single organization, the interest of the organization can be well modeled by a CP DG, since the productivity of the organization will be highest if the users are as dispersed as possible among the servers. However, the users' preferences may be more *selfish*: a user may prefer individually to use a resource with the fewest possible other users, regardless of the welfare of the rest of the group. Additionally, users' preferences may reflect some combination of individual and group welfare. These problems may be modeled with the class of *selfish dispersion games*.

Consider again the niche selection problem, in which each of n oligopoly producers wishes to occupy one of k different market niches. It may be the case that in addition to a general preference for dispersal (presumably to avoid competition) each producer has an *exogenous* preference for one of the niches; these preferences may or may not be aligned. For example, it may be that one of the market niches is larger and thus preferred by all producers. Alternatively, a producer may have competencies that suit it well for a particular niche. Note that the two agent case can be modeled by what one might call the *anti-battle-of-the-sexes game* in which a man and his ex-wife both wish to attend one of two parties, one of which is more desirable, but both prefer not to encounter each other (the reader familiar with the original BoS game will appreciate the humor). These problems can be modeled with the class of *partial dispersion games*, in which agents' preferences may align with either the dispersion ordering or with a set of exogenous preferences.

Learning Strategy Definitions

Now that we have defined a few interesting classes of dispersion games, let us consider the task of playing them in a repeated game setting. There are two perspectives we may adopt: that of the individual agent wishing to maximize his individual welfare, and that of a system designer wishing to implement a distributed algorithm for maximizing the group welfare. In the present research, we adopt the latter.

Let us begin with the problem of finding an MDO as quickly as possible in a weak CACP DG.[4] Note that this problem is trivial if implemented as a centralized algorithm. The problem is also trivial if implemented as a distributed algorithm in which agents are allowed unlimited communication. Thus we seek distributed algorithms that require no explicit communication between agents. Each algorithm takes the form of a set of identical learning rules for each

[4]Note that any mixed strategy equilibrium outcome is necessarily preference dominated by the pure strategy MDOs. For this reason, we henceforth disregard mixed strategy equilibria, and focus on the problem of finding one of the MDOs.

agent, each of which is a function mapping observed histories to distributions over actions.

Consider the most naive distributed algorithm. In each round, each agent selects an action randomly from the uniform distribution, stopping only when the outcome is an MDO. Note that this naive learning rule imposes very minimal information requirements on the agents: each agent must be informed only whether the outcome is an MDO. Unfortunately, the expected number of rounds until convergence to an MDO is

$$\frac{k^n}{MDO(n,k)}.$$

It is easy to see that for $k = n$ the expected time is $n^n/n!$, which is exponential in n.

We began by evaluating traditional learning rules from game theory and artificial intelligence. Game theory offers a plethora of options; we looked for simplicity and intuitive appropriateness. We considered both fictitious play (Brown 1951; Robinson 1951) and rational learning (Kalai & Lehrer 1993). Rational learning did not seem promising because of its dependence on the strategy space and initial beliefs of the agents. Thus we focused our attention on fictitious play.

In evaluating learning rules from artificial intelligence the decision was more straightforward. Recently there has been significant interest in the application of reinforcement learning to the problem of multi-agent system learning (Littman 1994; Claus & Boutilier 1998; Brafman & Tennenholtz 2000). We chose to implement and test the most common reinforcement learning algorithm: *Q-learning*.

Finally, we developed a few special purpose strategies to take advantage of the special structure of DGs.

Note that the different strategies we describe require agents to have access to different amounts of information about the outcome of each round as they play the game. At one extreme, agents might need only a Boolean value signifying whether or not the group has reached an MDO (this is all that is required for the naive strategy). At the other extreme, agents might need complete information about the outcome, including the action choices of each of the other agents.

Fictitious Play Learning

Fictitious play is a learning rule in which an agent assumes that each other agent is playing a fixed mixed strategy. The fictitious play agent uses counts of the actions selected by the other agents to estimate their mixed strategies and then at each round selects the action that has the highest expected value given these beliefs. Note that the fictitious play rule places very high information requirements on the agents. In order to update their beliefs, agents must have full knowledge of the outcome. Our implementation of fictitious play includes a few minor modifications to the basic rule.

One modification stems from the well known fact that agents using fictitious play may never converge to equilibrium play. Indeed our experiments show that fictitious play agents in CP DGs often generate play that oscillates within sets of outcomes, never reaching an MDO. This results from the agents' erroneous belief in the others' use of a fixed mixed strategy. To avoid this oscillation, we modify the fictitious play rule with stochastic perturbations of agents' beliefs as suggested by (Fudenberg & Levine 1998). In particular, we apply a uniform random variation of -1% to 1% on the expected reward of each action before selecting the agent's best response.

The other modifications were necessary to make the agents' computation within each round tractable for large numbers of agents. Calculating the expected value of each possible action at each round requires time that is exponential in n. To avoid this, we store the history of play as counts of observed outcomes rather than counts of each agents' actions. Also, instead of maintaining the entire history of play, we use a bounded memory of observed outcomes. The predicted joint mixed strategy of the other agents is then calculated by assuming the observed outcomes within memory are an unbiased sample. [5]

Reinforcement Learning

Reinforcement learning is a learning rule in which agents learn a mapping from states to actions (Kaelbling, Littman, & Moore 1996). We implemented the *Q-learning* algorithm with a Boltzman exploration policy. In Q-learning, agents learn the expected reward of performing an action in a given state. Our implementation of Q-learning includes a few minor modifications to the basic algorithm.

It is well known that the performance of Q-learning is extremely sensitive to a number of implementation details. First, the choice of a state space for the agent's Q-function is critical. We chose to use only a single state, so that in effect agents learn Q-values over actions only. Second, the selection of initial Q-values and temperature is critical. We found it best to set the initial Q-values to lie strictly within the range of the highest possible payoff (i.e., being alone) and the next highest (i.e., being with one other agent). We chose to parameterize the Boltzman learning function with an initial low temperature. These choices allow agents that initially choose a non-conflicting action to have high probability of continuing to play this action, and allow those that have collided with other agents to learn eventually the true value of the action and successively choose other actions until they find an action that does not conflict.

In our implementation we chose to give the agents a selfish reward instead of the global common-preference reward. The reward is a function of the number of other agents that choose the same action, not of the degree of dispersion of the group as a whole. This selfish reward has the advantage of giving the agents a signal that is more closely tied to the effects of their actions, while still being maximal for each agent when the agents have reached an MDO.

Specialized Strategies

The first specialized strategy that we propose is the *freeze strategy*. In the freeze strategy, an agent chooses actions

[5]The reader might be concerned that this approximation changes the convergence properties of the rule. Although this may be the case in some settings, in our experiments with small n no difference was observed from those using the full history.

randomly until the first time she is alone, at which point she continues to replay that action indefinitely, regardless of whether other agents choose the same action. It is easy to see that this strategy is guaranteed to converge in the limit, and that if it converges it will converge to an MDO. The freeze strategy also has the benefit of imposing very minimal information requirements: it requires an agent to know only how many agents chose the same action as she did in the previous round.

An improvement on the freeze strategy is the *basic simple strategy*, which was originally suggested by Alpern (2001). In this strategy, each agent begins by randomly choosing an action. Then, if no other agent chose the same action, she chooses the same action in the next round. Otherwise, she randomizes over the set of actions that were either unoccupied or selected by two or more agents. Note that the basic simple strategy requires that agents know only which actions had a single agent in them after each round.

Definition 8 (Basic Simple Strategy) *Given an outcome $o \in O$, an agent using the* basic simple strategy *will*

- *If $n_a^o = 1$, select action a with probability 1,*
- *Otherwise, select an action from the uniform distribution over actions $a' \in A$ for which $n_{a'}^o \neq 1$.*

We have extended the basic simple strategy to work in the broader class of games for which $n \neq k$.

Definition 9 (Extended Simple Strategy) *Given an outcome $o \in O$, an agent using the* extended simple strategy *will*

- *If $n_a^o \leq \lfloor n/k \rfloor$, select action a with probability 1,*
- *Otherwise, select action a with probability $\frac{n/k}{n_a^o}$ and with probability $(1 - \frac{n/k}{n_a^o})$ randomize over the actions a' for which $n_{a'}^o < \lceil n/k \rceil$.*

Unlike the basic strategy, the extended strategy does not assign uniform probabilities to all actions that were not chosen by the correct number of agents. Consider agents reacting to the outcome $\{2, 2, 0, 0\}$. In this case each agent is better off staying with probability 0.5 and jumping to each of the empty slots with probability 0.25, than randomizing uniformly over all four slots. The extended simple strategy can actually be further improved by assigning non-uniform probabilities to the actions a' for which $n_{a'}^o < \lceil n/k \rceil$. We have found empirically that the learning rule converges more rapidly when agents place more probability on the actions that have fewer other agents in them. Note that the extended simple strategy requires that agents know the number of agents selecting each action in the round; the identity of these agents is not required, however.

Experimental Results

The learning rules and strategies described above differ significantly in the empirical time to converge. In Figure 2 we plot as a function of n the convergence time of the learning rules in repeated symmetric weak DGs, averaged over 1000 trials. Table 1 summarizes the observed performance of each strategy (as well as the information requirements of each strategy). We discuss the performance of each of the strategies in turn.

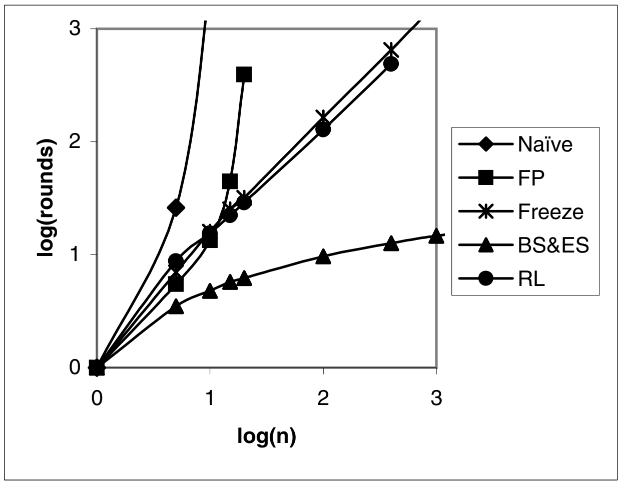


Figure 2: Log-log plot of the empirical performance of different strategies in symmetric CACP dispersion games.

Learning Rule	Information Requirements	Avg. Rounds to Converge ($f(n)$)
Naive	Whether MDO	EXP
FP	Full Information	EXP
RL	Num. in Own Action	POLY
Freeze	Num. in Own Action	LINEAR
BS & ES	Num. in All Actions	LOG

Table 1: Applicability of strategies to various classes of games with information requirements and estimated complexity class.

We begin with the learning rules. In our empirical tests we found that stochastic fictitious play always converged to an MDO. However, the number of rounds to converge was on average exponential in n. In our empirical tests of the reinforcement learning strategy we found that on average play converges to an MDO in a number of rounds that is linear in n. An interesting result is that for $n \neq k$, the algorithm didn't converge to a unique selection of actions for each agent, but rapidly adopted a set of mixed strategies for the agents resulting in average payoffs close to the optimal deterministic policy.

The specialized strategies generally exhibited better performance than the learning rules. Our empirical observations show that the number of rounds it takes for the freeze strategy to converge to an MDO is linear in n. Our empirical tests of both basic and extended simple strategies show that on average, play converges to an MDO in a number of steps that is logarithmic in the number of agents.[6]

[6]For $n > k$ certain ratios of n/k led consistently to superlogarithmic performance; slight modifications of the extended simple strategy were able to achieve logarithmic performance.

Discussion

In this paper we have introduced the class of DGs and defined several important subclasses that display interesting properties. We then investigated certain representative learning rules and tested their empirical behavior in DGs. In the future, we intend to continue this research in two primary directions.

First, we would like to further investigate some new types of DGs. We gave examples above of two classes of non-CP dispersion games that model common problems, but due to space limitations we were not able to define and characterize them in this paper. On a different note, we are also interested in a possible generalization of DGs which models the allocation of some quantity associated with the agents, such as skill or usage, to the different actions. We would like to define these classes of games formally, and explore learning rules that can solve them efficiently.

Second, we would like to continue the research on learning in DGs that we have begun in this paper. The learning rules we evaluated above are an initial exploration, and clearly many other learning techniques also deserve consideration. Additionally, we would like to complement the empirical work presented here with some analytical results. As a preliminary result, we can prove the following loose upper bound on the expected convergence time of the basic simple strategy.

Proposition 1 *In a repeated fully symmetric weak dispersion game with n agents and actions, in which all agents use the basic simple strategy, the expected number of rounds until convergence to an MDO is in $O(n)$.*

Informally, the proof is as follows. The probability that a particular agent chooses an action alone is $((n-1)/n)^{n-1}$, and so the expected number of rounds until she is alone is just $(n/(n-1))^{n-1}$. Because of the linearity of expectation, the expected number of rounds for all agents to find themselves alone must be no more than $n^n/(n-1)^{n-1}$, which is less than ne for all $n > 1$. Using similar techniques it is possible to show a quadratic bound on the expected convergence time of the freeze strategy.

Unfortunately, our empirical results show that the basic simple strategy converges in time that is logarithmic in n, and that the freeze strategy converges in linear time. This gap between our preliminary analysis and our empirical results begs future analytical work. Is it possible to show a tighter upper bound, for these learning rules or for others? Can we show a lower bound?

We would also like to better understand the optimality of learning rules. It is possible in principal to derive the optimal reactive learning rule for any finite number of agents using dynamic programming. Note that the optimal strategies obtained using this method are arbitrarily complex, however. For example, even upon reaching the simple outcome $\{2, 2, 0, 0\}$, an optimal reactive strategy for each agent chooses the same action with probability 0.5118 (not 0.5, as the extended simple strategy would dictate).

Dispersion games clearly play an important role in cooperative multiagent systems, and deserve much more discussion and scrutiny. We view the results of this paper as opening the door to substantial additional work on this exciting class of games.

References

Alpern, S. 2001. Spatial dispersion as a dynamic coordination problem. Technical report, The London School of Economics.

Arthur, B. 1994. Inductive reasoning and bounded rationality. *American Economic Association Papers* 84:406–411.

Azar, Y.; Broder, A. Z.; Karlin, A. R.; and Upfal, E. 2000. Balanced allocations. *SIAM Journal on Computing* 29(1):180–200.

Balch, T. 1998. Behavioral diversity in learning robot teams.

Brafman, R. I., and Tennenholtz, M. 2000. A near-optimal polynomial time algorithm for learning in certain classes of stochastic games. *Artificial Intelligence* 121(1-2):31–47.

Brown, G. 1951. Iterative solution of games by fictitious play. In *Activity Analysis of Production and Allocation*. New York: John Wiley and Sons.

Challet, D., and Zhang, Y. 1997. Emergence of cooperation and organization in an evolutionary game. *Physica A* 246:407.

Claus, C., and Boutilier, C. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI/IAAI*, 746–752.

Fudenberg, D., and Levine, D. K. 1998. *The Theory of Learning in Games*. Cambridge, MA: MIT Press.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. P. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4:237–285.

Kalai, E., and Lehrer, E. 1993. Rational learning leads to nash equilibrium. *Econometrica* 61(5):1019–1045.

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, 157–163. New Brunswick, NJ: Morgan Kaufmann.

Osborne, M., and Rubinstein, A. 1994. *A Course in Game Theory*. Cambridge, Massachusetts: MIT Press.

Robinson, J. 1951. An iterative method of solving a game. *Annals of Mathematics* 54:298–301.

Schelling, T. 1960. *The Strategy of Conflict*. Cambridge, Massachusetts: Harvard University Press.

Simple Search Methods for Finding a Nash Equilibrium

Ryan Porter and **Eugene Nudelman** and **Yoav Shoham**
Computer Science Department
Stanford University
Stanford, CA 94305
{rwporter,eugnud,shoham}@cs.stanford.edu

Abstract

We present two simple search methods for computing a sample Nash equilibrium in a normal-form game: one for 2-player games and one for n-player games. We test these algorithms on many classes of games, and show that they perform well against the state of the art– the Lemke-Howson algorithm for 2-player games, and Simplicial Subdivision and Govindan-Wilson for n-player games.

Introduction

Game theory has had a profound impact on multi-agent systems research, and indeed on computer science in general. Nash equilibrium (NE) is arguably the most important concept in game theory, and yet remarkably little is known about the problem of computing a sample NE in a normal-form game. All evidence points to this being a hard problem, but its precise complexity is unknown (Papadimitriou 2001).

At the same time, several algorithms have been proposed over the years for the problem. In this paper, three previous algorithms will be of particular interest. For 2-player games, the Lemke-Howson algorithm (Lemke & Howson 1964) is still the state of the art, despite being 40 years old. For n-player games, until recently the algorithm based on Simplicial Subdivision (van der Laan, Talman, & van der Heyden 1987) was the state of the art. Indeed, these two algorithms are the default ones implemented in Gambit (McKelvey, McLennan, & Turocy 2003), the best-known game theory software. Recently, a new algorithm, which we will refer to as Govindan-Wilson, was introduced by (Govindan & Wilson 2003) and extended and ef ciently implemented by (Blum, Shelton, & Koller 2003).

In a long version of this paper we provide more intuition behind each these methods. Here we simply note that they have surfaced as the most competitive algorithms for the respective class of games, and refer the reader to two thorough surveys on the topic (von Stengel 2002; McKelvey & McLennan 1996). Our goal in this paper is to demonstrate that for both of these classes of games (2-player, and n-player for $n > 2$) there exists a relatively simple, search-based method that performs very well in practice. For 2-player games, our algorithm performs substantially better than Lemke-Howson. For n-player games, our algorithm outperforms both Simplicial Subdivision and Govindan-Wilson.

The basic idea behind our search algorithms is simple. Recall that, while the general problem of computing a NE is a complementarity problem, computing whether there exists a NE with a *particular support*[2] for each player is a relatively easy feasibility program. Our algorithms explore the space of support pro les using a backtracking procedure to instantiate the support for each player separately. After each instantiation, they prune the search space by checking for actions in a support that are strictly dominated, given that the other agents will only play actions in their own supports.

Both algorithms order the search by giving precedence to supports of small size. Since it turns out that games drawn from classes that researchers have focused on in the past tend to have (at least one) NE with a very small support, our algorithms are often able to nd one quickly. Thus, this paper is as much about the properties of NE in games of interest as it is about novel algorithmic insights.

We emphasize, however, that we are not cheating in the selection of games on which we test. Past algorithms were tested almost exclusively on "random" games. We tested on these too (indeed, we will have more to say about how "random" games vary along at least one important dimension), but also on many other distributions (24 in total). To this end we use GAMUT, a recently introduced computational testbed for game theory (Nudelman et al. 2004). Our results are quite robust across all games tested.

The rest of the paper is organized as follows. After formulating the problem and the basis for searching over supports, we describe our two algorithms. The n-player algorithm is essentially a generalization of the 2-player algorithm, but we describe them separately, both because they differ slightly in the ordering of the search, and because the 2-player case admits a simpler description of the algorithm. Then, we describe our experimental setup, and separately present our results for 2-player and n-player games. In the nal section, we conclude and describe opportunities for future work.

[1]This work was supported in part by DARPA grant F30602-00-2-0598 and in part by the National Science Foundation under ITR IIS-0205633.

[2]The support speci es the pure strategies played with nonzero probability.

Notation

We consider nite, n-player, normal-form games $G = \langle N, (A_i), (u_i) \rangle$:

- $N = \{1, \ldots, n\}$ is the set of players.
- $A_i = \{a_{i1}, \ldots, a_{im_i}\}$ is the set of actions available to player i, where m_i is the number of available actions for that player. We will use a_i as a variable that takes on the value of a particular action a_{ij} of player i, and $a = (a_1, \ldots, a_n)$ to denote a pro le of actions, one for each player. Also, let $a_{-i} = (a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n)$ denote this same pro le excluding the action of player i, so that (a_i, a_{-i}) forms a complete pro le of actions. We will use similar notation for any pro le that contains an element for each player.
- $u_i : A_1 \times \ldots \times A_n \rightarrow \Re$ is the utility function for each player i. It maps a pro le of actions to a value.

Each player i selects a mixed strategy from the set $\mathcal{P}_i = \{p_i : A_i \rightarrow [0,1] | \sum_{a_i \in A_i} p_i(a_i) = 1\}$. A mixed strategy for a player speci es the probability distribution used to select the action that the player will play in the game. We will sometimes use a_i to denote the pure strategy in which $p_i(a_i) = 1$. The support of a mixed strategy p_i is the set of all actions $a_i \in A_i$ such that $p_i(a_i) > 0$. We will use $x = (x_1, \ldots, x_n)$ to denote a pro le of values that speci es the size of the support of each player.

Because agents use mixed strategies, u_i is extended to also denote the expected utility for player i for a strategy pro le $p = (p_1, \ldots, p_n)$: $u_i(p) = \sum_{a \in A} p(a) u_i(a)$, where $p(a) = \Pi_{i \in N} p_i(a_i)$.

The primary solution concept for a normal form game is that of Nash equilibrium. A mixed strategy pro le is a Nash equilibrium if no agent has incentive to unilaterally deviate.

De nition 1 *A strategy pro le $p^* \in \mathcal{P}$ is a Nash equilibrium if: $\forall i \in N, a_i \in A_i : \quad u_i(a_i, p^*_{-i}) \leq u_i(p^*_i, p^*_{-i})$*

Every nite, normal form game is guaranteed to have at least one Nash equilibrium (Nash 1950).

Searching Over Supports

The basis of our two algorithms is to search over the space of possible instantiations of the support $S_i \subseteq A_i$ for each player i. Given a support pro le as input, Feasibility Program 1, below, gives the formal description of a program for nding a Nash equilibrium p consistent with S (if such an strategy pro le exists).[3] In this program, v_i corresponds to the expected utility of player i in an equilibrium. The rst two classes of constraints require that each player must be indifferent between all actions within his support, and must not strictly prefer an action outside of his support. These imply that no player can deviate to a pure strategy that improves his expected utility, which is exactly the condition for the strategy pro le to be a Nash equilibrium.

[3]We note that the use of Feasibility Program 1 is not novel– it was used by (Dickhaut & Kaplan 1991) in an algorithm which enumerated all support pro les in order to nd all Nash equilibria.

Because $p(a_{-i}) = \prod_{j \neq i} p_j(a_j)$, this program is linear for $n = 2$ and nonlinear for all $n > 2$. Note that, strictly speaking, we do not require that each action $a_i \in S_i$ be in the support, because it is allowed to be played with zero probability. However, player i must still be indifferent between action a_i and each other action $a'_i \in S_i$.

Feasibility Program 1

Input: $S = (S_1, \ldots, S_n)$, a support pro le

Output: NE p, if there exists both a strategy pro le $p = (p_1, \ldots, p_n)$ and a value pro le $v = (v_1, \ldots, v_n)$ s.t.:

$$\begin{aligned}
\forall i \in N, a_i \in S_i : &\quad \sum_{a_{-i} \in S_{-i}} p(a_{-i}) u_i(a_i, a_{-i}) = v_i \\
\forall i \in N, a_i \notin S_i : &\quad \sum_{a_{-i} \in S_{-i}} p(a_{-i}) u_i(a_i, a_{-i}) \leq v_i \\
\forall i \in N : &\quad \sum_{a_i \in S_i} p_i(a_i) = 1 \\
\forall i \in N, a_i \in S_i : &\quad p_i(a_i) \geq 0 \\
\forall i \in N, a_i \notin S_i : &\quad p_i(a_i) = 0
\end{aligned}$$

Algorithm for Two-Player Games

In this section we describe Algorithm 1, our 2-player algorithm for searching the space of supports. There are three keys to the ef cienc y of this algorithm. The rst two are the factors used to order the search space. Speci cally , Algorithm 1 considers every possible support size pro le separately, favoring support sizes that are balanced and small. The motivation behind these choices comes from work such as (McLennan & Berg 2002), which analyzes the theoretical properties of the NE of games drawn from a particular distribution. Speci cally , for n-player games, the payoffs for an action pro le are determined by drawing a point uniformly at random in a unit sphere. Under this distribution, for $n = 2$, the probability that there exists a NE consistent with a particular support pro le varies inversely with the size of the supports, and is zero for unbalanced support pro les.

The third key to Algorithm 1 is that it separately instantiates each players' support, making use of what we will call "conditional (strict) dominance" to prune the search space.

De nition 2 *An action $a_i \in A_i$ is conditionally dominated, given a pro le of sets of available actions $R_{-i} \subseteq A_{-i}$ for the remaining agents, if the following condition holds: $\exists a'_i \in A_i \; \forall a_{-i} \in R_{-i} : \quad u_i(a_i, a_{-i}) < u_i(a'_i, a_{-i})$.*

The preference for small support sizes ampli es the advantages of checking for conditional dominance. For example, after instantiating a support of size two for the rst player, it will often be the case that many of the second player's actions are pruned, because only two inequalities must hold for one action to conditionally dominate another.

Pseudo-code for Algorithm 1 is given below. Note that this algorithm is complete, because it considers all support size pro les, and because it only prunes actions that are *strictly* dominated.

Algorithm 1

```
for all support size pro les $x = (x_1, x_2)$, sorted in increasing order of, rst, $|x_1 - x_2|$ and, second, $(x_1 + x_2)$ do
    for all $S_1 \subseteq A_1$ s.t. $|S_1| = x_1$ do
        $A'_2 \leftarrow \{a_2 \in A_2$ not cond. dominated, given $S_1\}$
        if $\nexists a_1 \in S_1$ cond. dominated, given $A'_2$ then
            for all $S_2 \subseteq A'_2$ s.t. $|S_2| = x_2$ do
                if $\nexists a_1 \in S_1$ cond. dominated, given $S_2$ then
                    if Feasibility Program 1 is satis able for $S = (S_1, S_2)$ then
                        Return the found NE $p$
```

Algorithm for N-Player Games

Algorithm 1 can be interpreted as using the general backtracking algorithm (see, e.g., (Dechter 2003)) to solve a constraint satisfaction problem (CSP) for each support size pro le. The variables in each CSP are the supports S_i, and the domain of each S_i is the set of supports of size x_i. While the single constraint is that there must exist a solution to Feasibility Program 1, an extraneous, but easier to check, set of constraints is that no agent plays a conditionally dominated action. The removal of conditionally dominated strategies by Algorithm 1 is similar to using the AC-1 to enforce arc-consistency with respect to these constraints. We use this interpretation to generalize Algorithm 1 for the n-player case. Pseudo-code for Algorithm 2 and its two procedures, Recursive-Backtracking and Iterated Removal of Strictly Dominated Strategies (IRSDS) are given below.[4]

IRSDS takes as input a domain for each player's support. For each agent whose support has been instantiated, the domain contains only that instantiated support, while for each other agent i it contains all supports of size x_i that were not eliminated in a previous call to this procedure. On each pass of the *repeat-until* loop, every action found in at least one support of a player's domain is checked for conditional domination. If a domain becomes empty after the removal of a conditionally dominated action, then the current instantiations of the Recursive-Backtracking are inconsistent, and IRSDS returns *failure*. Because the removal of an action can lead to further domain reductions for other agents, IRSDS repeats until it either returns *failure* or iterates through all actions of all players without nding a dominated action.

Finally, we note that Algorithm 2 is not a strict generalization of Algorithm 1, because it orders the support size pro les rst by size, and then by a measure of balance. The reason for the change is that balance (while still signi cant) is less important for $n > 2$ than it is for $n = 2$. For example, under the model of (McLennan & Berg 2002), for $n > 2$, the probability of the existence of a NE consistent with a particular support pro le is no longer zero when the support pro le is unbalanced.

[4] Even though our implementation of the backtracking procedure is iterative, for simplicity we present it here in its equivalent, recursive form. Also, the reader familiar with CSPs will recognize that we have employed very basic algorithms for backtracking and for enforcing arc consistency, and we return to this point in the conclusion.

Algorithm 2

```
for all $x = (x_1, \ldots, x_n)$, sorted in increasing order of, rst, $\sum_i x_i$ and, second, $max_{i,j}(x_i - x_j)$ do
    $\forall i : S_i \leftarrow NULL$                //uninstantiated supports
    $\forall i : D_i \leftarrow \{S_i \subseteq A_i : |S_i| = x_i\}$   //domain of supports
    if Recursive-Backtracking($S, D, 1$) returns a NE $p$ then
        Return $p$
```

Procedure 1 Recursive-Backtracking

```
Input: $S = (S_1, \ldots, S_n)$: a pro le of supports
       $D = (D_1, \ldots, D_n)$: a pro le of domains
       $i$: index of next support to instantiate
Output: A Nash equilibrium $p$, or $failure$
if $i = n + 1$ then
    if Feasibility Program 1 is satis able for $S$ then
        Return the found NE $p$
    else
        Return $failure$
else
    for all $d_i \in D_i$ do
        $S_i \leftarrow d_i$
        $D_i \leftarrow D_i - \{d_i\}$
        if IRSDS(($\{S_1\}, \ldots, \{S_i\}, D_{i+1}, \ldots, D_n$)) succeeds then
            if Recursive-Backtracking($S, D, i + 1$) returns NE $p$ then
                Return $p$
    Return $failure$
```

Procedure 2 Iterated Removal of Strictly Dominated Strategies (IRSDS)

```
Input: $D = (D_1, \ldots, D_n)$: pro le of domains
Output: Updated domains, or $failure$
repeat
    $changed \leftarrow false$
    for all $i \in N$ do
        for all $a_i \in d_i \in D_i$ do
            for all $a'_i \in A_i$ do
                if $\forall a_{-i} \in d_{-i} \in D_{-i}, u_i(a_i, a_{-i}) < u_i(a'_i, a_{-i})$ then
                    $D_i \leftarrow D_i - \{d_i \in D_i : a_i \in d_i\}$
                    $changed \leftarrow true$
                    if $D_i = \emptyset$ then
                        return $failure$
until $changed = false$
return $D$
```

D1	Bertrand Oligopoly	D2	Bidirectional LEG, Complete Graph
D3	Bidirectional LEG, Random Graph	D4	Bidirectional LEG, Star Graph
D5	Covariance Game: $\rho = 0.9$	D6	Cov. Game: $\rho \in [-1/(N-1), 1]$
D7	Covariance Game: $\rho = 0$	D8	Dispersion Game
D9	Graphical Game, Random Graph	D10	Graphical Game, Road Graph
D11	Graphical Game, Star Graph	D12	Graphical Game, Small-World
D13	Minimum Effort Game	D14	Polymatrix Game, Complete Graph
D15	Polymatrix Game, Random Graph	D16	Polymatrix Game, Road Graph
D17	PolymatrixGame, Small-World	D18	Uniformly Random Game
D19	Travelers Dilemma	D20	Uniform LEG, Complete Graph
D21	Uniform LEG, Random Graph	D22	Uniform LEG, Star Graph
D23	Location Game	D24	War Of Attrition

Table 1: Descriptions of GAMUT distributions.

Experimental Results

To evaluate the performance of our algorithms we ran several sets of experiments. All games were generated by GAMUT (Nudelman et al. 2004), a test-suite that is capable of generating games from a wide variety of classes of games found in the literature. Table 1 provides a brief description of the subset of distributions on which we tested.

A distribution of particular importance is the one most commonly tested on in previous work: D18, the "Uniformly Random Game", in which every payoff in the game is drawn independently from an identical uniform distribution. Also important are distributions D5, D6, and D7, which fall under a "Covariance Game" model studied by (Rinott & Scarsini 2000), in which the payoffs for the n agents for each action pro le are drawn from a multivariate normal distribution in which the covariance ρ between the payoffs of each pair of agents is identical. When $\rho = 1$, the game is common-payoff, while $\rho = \frac{-1}{N-1}$ yields minimal correlation, which occurs in zero-sum games. Thus, by altering ρ, we can smoothly transition between these two extreme classes of games.

Our experiments were executed on a cluster of 12 dual-processor, 2.4GHz Pentium machines, running Linux 2.4.20. We capped runs for all algorithms at 1800 seconds. When describing the statistics used to evaluate the algorithms, we will use "unconditional" to refer to the value of the statistic when timeouts are counted as 1800 seconds, and "conditional" to refer to its value excluding timeouts.

When $n = 2$, we solved Feasibility Program 1 using CPLEX 8.0's callable library. For $n > 2$, because the program is nonlinear, we instead solved each instance of the program by executing AMPL, using MINOS as the underlying optimization package. Obviously, we could substitute in any nonlinear solver; and, since a large fraction of our running time is spent on AMPL and MINOS, doing so would greatly affect the overall running time.

Before presenting the empirical results, we note that a comparison of the worst-case running times of our two algorithms and the three we tested against does not distinguish between them, since there exist inputs for each which lead to exponential time.

Results for Two-Player Games

In the rst set of experiments, we compared the performance of Algorithm 1 to that of Lemke-Howson (implemented in Gambit, which added the preprocessing step of iterated removal of weakly dominated strategies) on 2-player, 300-action games drawn from 24 of GAMUT's 2-player distributions. Both algorithms were executed on 100 games drawn from each distribution. The time is measured in seconds and plotted on a logarithmic scale.

Figure 1(a) compares the unconditional median runtimes of the two algorithms, and shows that Algorithm 1 performs better on all distributions.[5] However, this does not tell the whole story. For many distributions, it simply re ects the fact that there is a greater than 50% chance that the distribution will generate a game with a pure strategy NE, which our algorithm will then nd quickly. Two other important statistics are the percentage of instances solved (Figure 1(b)), and the average runtime conditional on solving the instance (Figure 1(c)). Here, we see that Algorithm 1 completes far more instances on several distributions, and solves fewer on just a single distribution (6 fewer, on D23). Additionally, even on distributions for which we solve far more games, our conditional average runtime is 1 to 2 orders of magnitude smaller.

Clearly, the hardest distribution for our algorithm is D6, which consists of "Covariance Games" in which the covariance ρ is drawn uniformly at random from the range $[-1, 1]$. In fact, neither Algorithm 1 nor Lemke-Howson solved any of the games in another "Covariance Game" distribution in which $\rho = -0.9$, and these results were omitted from the graphs, because the conditional average is unde ned for these results. On the other hand, for the distribution "CovarianceGame-Pos" (D5), in which $\rho = 0.9$, both algorithms perform well.

To further investigate this continuum, we sampled 300 values for ρ in the range $[-1, 1]$, with heavier sampling in the transition region and at zero. For each such game, we plotted a point for the runtime of both Algorithm 1 and Lemke-Howson in Figure 1(d).[6] The theoretical results of (Rinott & Scarsini 2000) suggest that the games with lower covariance should be more dif cult for Algorithm 1, because they are less likely to have a pure strategy Nash equilibrium. Nevertheless, it is interesting to note the sharpness of the transition that occurs in the $[-0.3, 0]$ interval. More surprisingly, a similarly sharp transition also occurs for Lemke-Howson, despite the fact that the two algorithms operate in unrelated ways. Finally, it is important to note that the transition region for Lemke-Howson is shifted to the right by approximately 0.3, and that, on instances in the easy region for both algorithms, Algorithm 1 is still an order of magnitude faster.

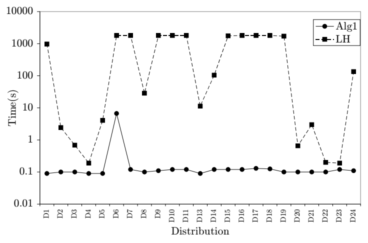
In the third set of experiments we explore the scaling behavior of both algorithms on the "Uniformly Random Game" distribution (D18), as the number of actions increases from 100 to 1000. For each multiple of 100, we generated 20 games. Because space constraints preclude an analysis similar to that of Figures 1(a) through 1(c), we instead plot in Figure 1(e) the *unconditional* average runtime over 20 instances for each data size, with a timeout counted as 1800s. While Lemke-Howson failed to solve any game with more than 600 actions and timed out on some 100-action games, Algorithm 1 solved all instances, and, without the help of cutoff times, still had an advantage of 2 orders of magnitude at 1000 actions.

Results for N-Player Games

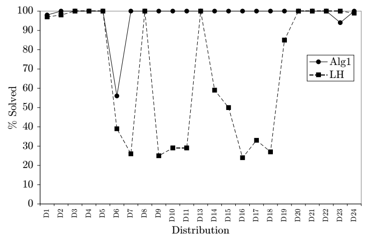
In the next set of experiments we compare Algorithm 2 to Govindan-Wilson and Simplicial Subdivision (which was implemented in Gambit, and thus combined with iterated removal of weakly dominated strategies). First, to compare performance on a x ed problem size we tested on 6-player,

[5]Obviously, the lines connecting data points across distributions for a particular algorithm are meaningless– they were only added to make the graph easier to read.

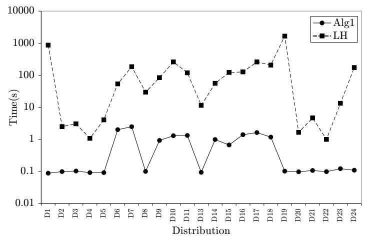
[6]The capped instances for Algorithm 1 were perturbed slightly upward on the graph for clarity.



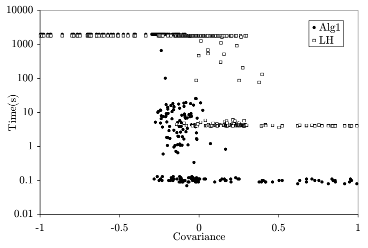
(a) Unconditional median runtime



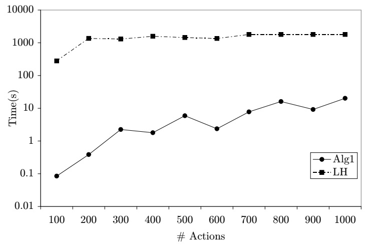
(b) Percentage solved



(c) Average time on solved instances

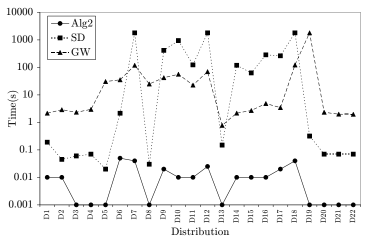


(d) Runtime vs. Covariance

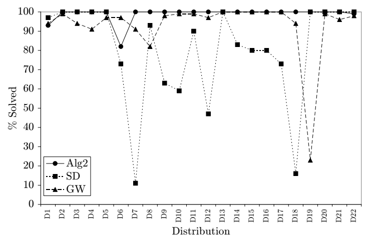


(e) Unconditional average vs. Actions

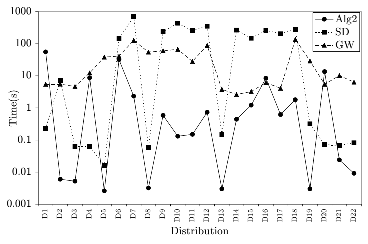
Figure 1: Comparison of Algorithm 1 and Lemke-Howson on 2-player games. Sub gures (a)-(d) are for 300-action games.



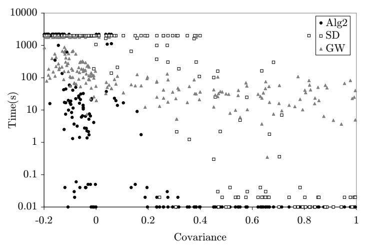
(a) Unconditional median runtimes



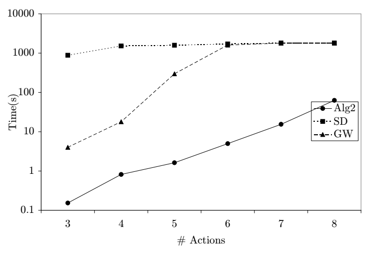
(b) Percentage solved



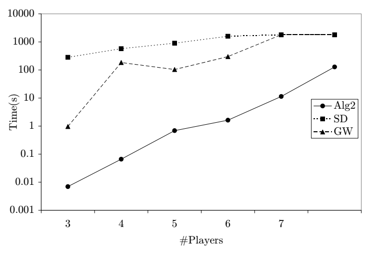
(c) Average time on solved instances



(d) Runtime vs. Covariance



(e) Unconditional average runtime vs. Actions, on 6-player games



(f) Unconditional average runtime vs. Players, on 5-action games

Figure 2: Comparison of Algorithm 2, Simplicial Subdivision, and Govindan-Wilson. Sub gures (a)-(d) are for 6-player, 5-action games.

5-action games drawn from 22 of GAMUT's n-player distributions.[7] While the numbers of players and actions appear small, note that these games have 15625 outcomes and 93750 payoffs. Once again, Figures 2(a), 2(b), and 2(c) show unconditional median runtime, percentage of instances solved, and conditional average runtime, respectively. Algorithm 2 has a very low unconditional median runtime, for the same reason that Algorithm 1 did for two-player games, and outperforms both other algorithms on all distributions. While this dominance does not extend to the other two metrics, the comparison still favors Algorithm 2.

We again investigate the relationship between ρ and the hardness of games under the "Covariance Game" model. For general n-player games, minimal correlation under this model occurs when $\rho = -\frac{1}{n-1}$. Thus, we can only study the range $[-0.2, 1]$ for 6-player games. Figure 2(d) shows the results for 6-player 5-action games. Algorithm 2, over the range $[-0.1, 0]$, experiences a transition in hardness that is even sharper than that of Algorithm 1. Simplicial Subdivision also undergoes a transition, which is not as sharp, that begins at a much larger value of ρ (around 0.4). However, the running time of Govindan-Wilson is only slightly affected by the covariance, as it neither suffers as much for small values of ρ nor bene ts as much from large values.

Finally, Figures 2(e) and 2(f) compare the scaling behavior (in terms of unconditional average runtimes) of the three algorithms: the former holds the number of players constant at 6 and varies the number of actions from 3 to 8, while the latter holds the number of actions constant at 5, and varies the number of players from 3 to 8. In both experiments, both Simplicial Subdivision and Govindan-Wilson solve no instances for the largest two sizes, while Algorithm 2 still nds a solution for most games.

Conclusion and Future Work

In this paper, we presented two algorithms for nding a sample Nash equilibrium. Both use backtracking approaches (augmented with pruning) to search the space of support pro les, favoring supports that are small and balanced. Both also outperform the current state of the art.

The most dif cult games we encountered came from the "Covariance Game" model, as the covariance approaches its minimal value, and this is a natural target for future algorithm development. We expect these games to be hard in general, because, empirically, we found that as the covariance decreases, the number of equilibria decreases, and the equilibria that do exist are more likely to have support sizes near one half of the number of actions, which is the support size with the largest number of supports.

One direction for future work is to employ more sophisticated CSP techniques. The main goal of this paper was to show that our general search method performs well in practice, and there are many other CSP search and inference strategies which may improve its ef cienc y. Another promising direction to explore is local search, in which the state space is the set of all possible supports, and the available moves are to add or delete an action from the support of a player. While the fact that no equilibrium exists for a particular support does not give any guidance as to which neighboring support to explore next, one could use a relaxation of Feasibility Program 1 that penalizes infeasibility through an objective function. More generally, our results show that AI techniques can be successfully applied to this problem, and we have only scratched the surface of possibilities along this direction.

[7] Two distributions from the tests of 2-player games are missing here, due to the fact that they do not naturally generalize to more than 2 players.

References

Blum, B.; Shelton, C. R.; and Koller, D. 2003. A continuation method for Nash equilibria in structured games. In *IJCAI-03*.

Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.

Dickhaut, J., and Kaplan, T. 1991. A program for nding Nash equilibria. *The Mathematica Journal* 87–93.

Govindan, S., and Wilson, R. 2003. A global newton method to compute Nash equilibria. In *Journal of Economic Theory*.

Lemke, C., and Howson, J. 1964. Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics* 12:413–423.

McKelvey, R., and McLennan, A. 1996. Computation of equilibria in nite games. In H. Amman, D. Kendrick, J. R., ed., *Handbook of Computational Economics*, volume I. Elsevier. 87–142.

McKelvey, R.; McLennan, A.; and Turocy, T. 2003. Gambit: Software tools for game theory, version 0.97.0.5. Available at http://econweb.tamu.edu/gambit/.

McLennan, A., and Berg, J. 2002. The asymptotic expected number of Nash equilibria of two player normal form games. Mimeo, University of Minnesota.

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.

Nudelman, E.; Wortman, J.; Shoham, Y.; and Leyton-Brown, K. 2004. Run the GAMUT: A comprehensive approach to evaluating game-theoretic algorithms. In *AAMAS-04*.

Papadimitriou, C. 2001. Algorithms, games, and the internet. In *STOC-01*, 749–753.

Rinott, Y., and Scarsini, M. 2000. On the number of pure strategy Nash equilibria in random games. *Games and Economic Behavior* 33:274–293.

van der Laan, G.; Talman, A.; and van der Heyden, L. 1987. Simplicial variable dimension algorithms for solving the nonlinear complementarity problem on a product of unit simplices using a general labelling. *Mathematics of Operations Research*.

von Stengel, B. 2002. Computing equilibria for two-person games. In Aumann, R., and Hart, S., eds., *Handbook of Game Theory*, volume 3. North-Holland. chapter 45, 1723–1759.

Run the GAMUT: A Comprehensive Approach to Evaluating Game-Theoretic Algorithms

Eugene Nudelman Jennifer Wortman Yoav Shoham
{eugnud;jwortman;shoham}@cs.stanford.edu Stanford University, Stanford, CA 94305
Kevin Leyton-Brown
kevinlb@cs.ubc.ca University of British Columbia, Vancouver, BC V6T 1Z4

Abstract

We present GAMUT[1], a suite of game generators designed for testing game-theoretic algorithms. We explain why such a generator is necessary, offer a way of visualizing relationships between the sets of games supported by GAMUT, and give an overview of GAMUT's architecture. We highlight the importance of using comprehensive test data by benchmarking existing algorithms. We show surprisingly large variation in algorithm performance across different sets of games for two widely-studied problems: computing Nash equilibria and multiagent learning in repeated games.[2]

1. Introduction

Researchers in multiagent systems have become increasingly interested in game theory as a modeling tool. This has led to growing interest in computational problems associated with game-theoretic domains. Two such problems are computing Nash equilibria and learning to achieve good payoffs in repeated games. It is often difficult to offer theoretical guarantees about such algorithms' performance: the computational complexity of many algorithms for computing Nash remains an interesting open problem [14], and there is rarely anything that can be proven about the sort of performance a learning algorithm will achieve without making reference to the game it will play or the opponents it will face. For these sorts of reasons, researchers needing to evaluate algorithms for game-theoretic problems often choose to perform empirical tests.

One general lesson that has been learned by researchers working in a wide variety of different domains is that an algorithm's performance can vary substantially across different "reasonable" distributions of problem instances, even when problem size is held constant [9]. When we examine the empirical tests that have been performed on algorithms that take games as their inputs, we find that they have typically been small-scale and involved very particular choices of games. Such tests can be appropriate for limited proofs-of-concept, but cannot say much about an algorithm's expected performance in new domains. For this, a comprehensive body of test data is required.

It is not obvious that a library of games should be difficult to construct. After all, games (if we think for the moment about normal-form representations) are simply matrices with one dimension indexed by action for each player, and one further dimension indexed by player. We can thus generate games by taking the number of players and of actions for each player as parameters, and populate the corresponding matrix with real numbers generated uniformly at random. Is anything further required?

We set out to answer this question by studying sets of games that have been identified as interesting by computer scientists, game theorists, economists, political scientists and others over the past 50 years. Our attempt to get a sense of this huge literature led us to look at several hundred books and papers, and to extract one or more sets of games from more than a hundred sources. To our surprise, we discovered two things.

First, for *every one* of the sets of games that we encountered, the technique described above would generate a game from that set with probability zero. More formally, all of these sets are *non-generic* with respect to the uniform sampling procedure. It is very significant to find that an unbiased method of generating games has only an infinitesimal chance of generating any of these games that have been considered realistic or interesting. Since we know that algorithm performance can depend heavily on the choice of test data, it would be unreasonable to extrapolate from an algorithm's performance on random test data to its expected performance on real-world problems. It seems that test data for games must take the form of a patchwork of generators of

1 Available at `http://gamut.stanford.edu`
2 This work was supported by NSF grant IIS-0205633 and DARPA grant F30602-00-2-0598.

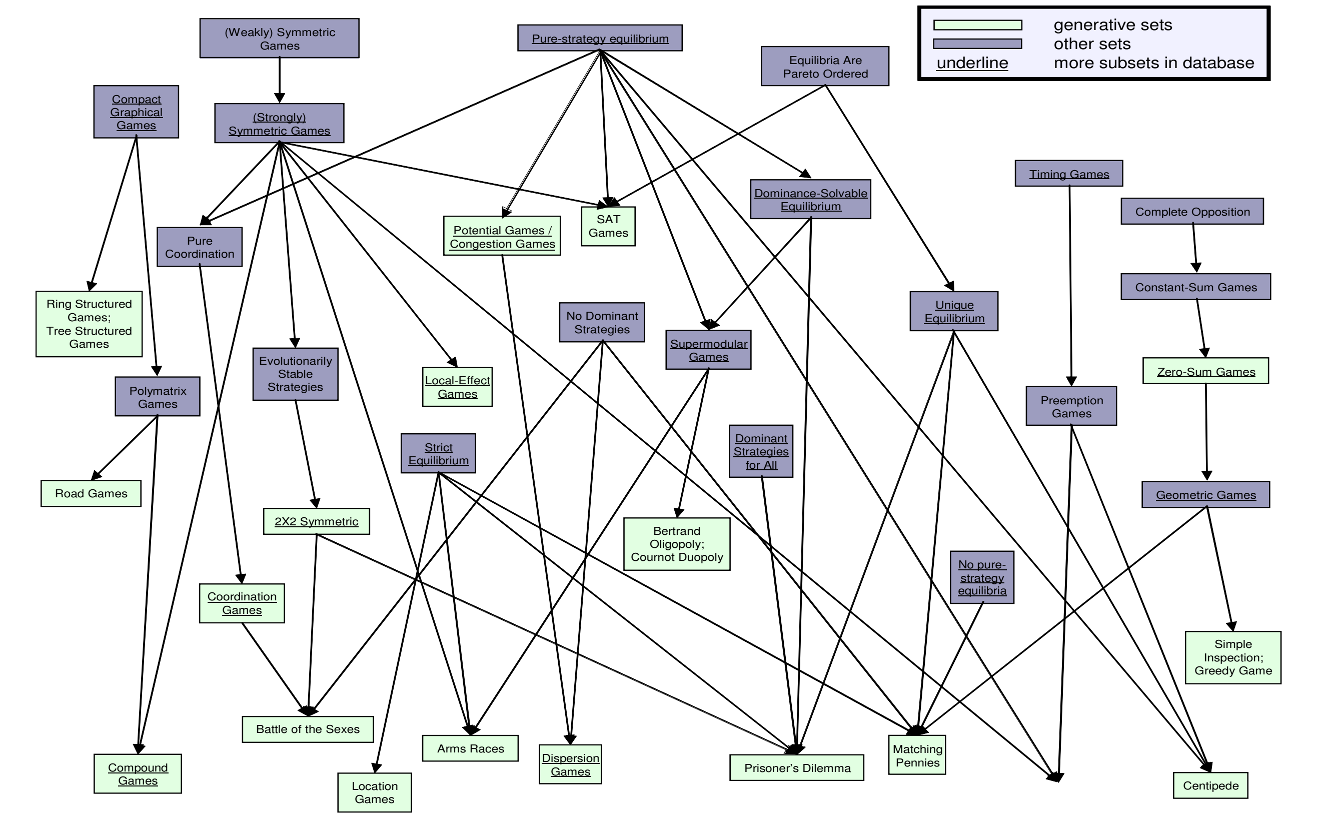


Figure 1. GAMUT Taxonomy (Partial)

different sets of games.

Second, we were surprised to find very little work that aimed to understand, taxonomize or even enumerate non-generic games in a holistic or integrative way. We came across work on understanding *generic* games [7], and found a complete taxonomy of two-player two-action games [15]. Otherwise, work that we encountered tended to fall into one or both of two camps. Some work aimed to describe and characterize particular sets of games that were proposed as reasonable models of real-world strategic situations or that presented interesting theoretical problems. Second, researchers proposed novel representations of games, explicitly or implicitly identifying sets of games that could be specified compactly in these representations.

In this paper we aim to fill this gap: to identify interesting sets of non-generic games comprehensively and with as little bias as possible. In the next section we describe this effort, highlighting relationships between different sets of games we encountered in our literature search and describing issues that arose in the identification of game generation algorithms. In section 3 we give experimental proof that a comprehensive test suite is required for the evaluation of game-theoretic algorithms. For our two example problems, computing Nash equilibria and learning in repeated games, we show that performance for different algorithms varies dramatically across different sets of games even when the size of the game is held constant, and that performance on random games can be a bad predictor of performance on other games. Finally, in the appendix, we briefly describe GAMUT's architecture and implementation, including discussion of how new games may easily be added.

2. GAMUT

For the initial version of GAMUT we considered only games whose normal-form representations can be comfortably stored in a computer. Note that this restriction does not rule out games that are presented in a more compact representation such as extensive form or graphical games; it only rules out *large* examples of such games. It also rules out games with infinite numbers of agents and/or of actions and Bayesian games. We make no requirement that games must *actually* be stored in normal form; in fact, GAMUT supports a wide array of representations (see the appendix). Some are *complete* (able to represent any game) while other *incomplete* representations support only certain sets of games. We will say that a given representation describes a set of games *compactly* if its descriptions of games in the set are exponentially shorter than the games' descriptions in normal form.

In total we identified 122 interesting sets of games in our literature search, and we were able to find finite time generative procedures for 71. These generative sets ranged from specific two-by-two matrix games with little variation (e.g., Chicken) to broad classes extensible in both number of players and number of actions (e.g., games that can be encoded compactly in the Graphical Game representation).

Arms Race	Grab the Dollar	Polymatrix Game
Battle of the Sexes	Graphical Game	Prisoner's Dilemma
Bertrand Oligopoly	Greedy Game	Random Games
Bidirectional LEG	Guess 2/3 Average	Rapoport's Distribution
Chicken	Hawk and Dove	Rock, Paper, Scissors
Collaboration Game	Local-Effect Game	Shapley's Game
Compound Game	Location Game	Simple Inspection Game
Congestion Game	Majority Voting	Traveler's Dilemma
Coordination Game	Matching Pennies	Uniform LEG
Cournot Duopoly	Minimum Effort Game	War of Attrition
Covariant Game	N-Player Chicken	Zero Sum Game
Dispersion Game	N-Player Pris Dilemma	

Table 1. Game Generators in GAMUT

2.1. The Games

To try to understand the relationships between these different sets of non-generic games, we set out to relate them taxonomically. We settled on identifying subset relationships between the different sets of games. Our taxonomy is too large to show in full, but a fragment of it is shown in Figure 1. To illustrate the sort of information that can be conveyed by this figure, we can see that all Dispersion Games [6] are Congestion Games [17] and that all Congestion Games have pure-strategy equilibria.

Besides providing some insight into the breadth of generators included in GAMUT and the relationships between them, our taxonomy also serves a more practical purpose: allowing the quick and intuitive selection of a set of generators. If GAMUT is directed to generate a game from a set that does not have a generator (e.g., supermodular games [13]; games having unique equilibria) it chooses uniformly at random among the generative descendants of the set and then generates a game from the chosen set. GAMUT also supports generating games that belong to multiple intersecting sets (e.g., symmetric games having pure-strategy equilibria); in this case GAMUT chooses uniformly at random among the generative sets that are descendants of all the named sets.

The data we collected in our literature search—including bibliographic references, pseudo-code for generating games and taxonomic relationships between games—will be useful to some researchers in its own right. We have gathered this information into a database which is publicly available from `http://gamut.stanford.edu` as part of the GAMUT release. Besides providing more information about references than we can fit into a conference-length paper, this database also allows users to navigate according to subset/superset relationships and to perform searches.

2.2. The Generators

Roughly speaking, the sets of games that we enumerated in the taxonomy can be partitioned into two classes, reflected by different colored nodes in Figure 1. For some sets we were able to come up with an efficient algorithmic procedure that can, in finite time, produce a sample game from that set, and that has the ability to produce any game from that set. We call such sets *generative*. For others, we could find no reasonable procedure. One might consider a rejection sampling approach that would generate games at random and then test whether they belong to a given set S. However, if S is non-generic—which is true for most of our sets, as discussed above—such a procedure would fail to produce a sample game in any finite amount of time. Thus, we do not consider such procedures as generators.

Cataloging the relationships among sets of games and identifying generators prepared us for our next task, creating game generators. The wrinkle was that generative algorithms were rarely described explicitly in the literature. While in most cases coming up with an algorithm was straightforward, we did encounter several interesting issues.

Sometimes an author defined a game too narrowly for our purposes. Many traditional games (e.g., Prisoner's Dilemma) are often defined in terms of precise payoffs. Since our goal was to construct a generator capable of producing an infinite number of games belonging to the same set, we had to generalize these games. In the case of Prisoner's Dilemma, we can generate any game

$$\begin{pmatrix} R, R & S, T \\ T, S & P, P \end{pmatrix}$$

which satisfies $T > R > P > S$ and $R > (S+T)/2$. (The latter condition ensures that all three of the non-equilibrium outcomes are Pareto optimal.) Thus, an algorithm for generating an instance of Prisoner's Dilemma reduces to generating four numbers that satisfy the given constraints. There is one subtlety involved with this approach to generalizing games. It is a well-known fact that a positive affine transformation of payoffs does not change strategic situation modeled by the game. It is also a common practice to normalize payoffs to some standard range before reasoning about games. We ensure that no generator ever generates instances that differ only by a positive affine transformation of payoffs.

In other cases the definition of a set was too broad, and thus had to be restricted. In many cases, this could be achieved via an appropriate parametrization. An interesting example of this is the set of Polymatrix Games [5]. These are n-player games with a very special payoff structure: every pair of agents plays a (potentially different) 2-player game between them, and each agent's utility is the sum of all of his payoffs. The caveat, however, is that the agent must play the *same* action in all of his two-player games. We realized that these games, though originally studied for their computational properties, could be generalized and used essentially as a compact representation for games in which each agent only plays two-player games against some *subset* of the other agents. This led to a natural parametrization of polymatrix games with graphs. Nodes of the graph now represent agents, and edges are labeled with different 2-player games.[3] Thus, though we still can sample from the set of all polymatrix games using a complete graph, we are now able also to focus on more specific and, thus, even more structured subsets.

Sometimes we encountered purely algorithmic difficulties. For example, in order to implement geometric games [18] we needed data structures capable of representing and performing operations on abstract sets (such as finding intersection, or enumerating subsets).

In some cases one parameterized generator was able to generate games from many different sets. For example, we implemented a single generator based on work by Rapoport [15] which demonstrated that there are only 85 strategically different 2x2 games, and so did not need to implement generators for individual 2x2 games mentioned in the literature. We did elect to create separate generators for several very common games (e.g., Matching Pennies; Hawk and Dove). We also used our taxonomy to identify similar sets of games, and either implemented them with the same generator or allowed their separate generators to benefit from sharing common algorithms and data structures. In the end we built 35 parameterized generators to support all of the generative sets in our taxonomy; these are listed in Table 1.

The process of writing generators presented us with a nontrivial software engineering task in creating a coherent and easily-extensible software framework. Once the framework was in place, incrementally adding new generators became easy. Some of these implementation details are described in the Appendix.

3. Running the GAMUT

At the beginning of this paper we claimed that it is necessary to evaluate game-theoretic algorithms on a wide range of distributions before empirical claims can be made about the algorithms' strengths and weaknesses. Of course, such a claim can only be substantiated after a test suite has been constructed. In this section we show that top algorithms for two computational problems in game theory do indeed exhibit dramatic variation across distributions, implying that small performance tests would be unreliable.

All our experiments were performed using a cluster of 12 dual-CPU 2.4GHz Xeon machines running Linux 2.4.20, and took about 120 CPU-days to run. We capped runs for all algorithms at 30 minutes (1800 seconds).

3.1. Computation of Nash Equilibria

One of the most interesting computational problems in game theory is computing Nash equilibria. All evidence suggests that this is a hard problem (e.g., [4, 3]), yet the precise complexity class into which the problem falls is unknown [14]. In this section we use GAMUT to evaluate three algorithms' empirical properties on this problem.

3.1.1. Experimental Setup The best-known game theory software package is Gambit [12], a collection of state-of-the-art algorithms. For two-player games the Lemke-Howson algorithm [8] is best and is used by default in Gambit. For n-player games Gambit uses an algorithm based on Simplicial Subdivision [19]. In both cases, Gambit performs iterative removal of dominated strategies as a preprocessing step. Govindan and Wilson [5] introduced an alternative algorithm based on a continuation method. We use a recent optimized implementation, the GameTracer package [1]. This work also included speedups for the Govindan-Wilson algorithm on the special cases of compact graphical games and MAIDs, but because we expanded all games to their full normal forms Govindan-Wilson did not benefit from these extensions in our experiments.

One factor that can have a significant effect on an algorithm's runtime is the size of its input. Since our goal was to investigate the extent to which runtimes vary as the result of differences between distributions, we studied fixed-size games. To make sure that our findings were not artifacts of any particular problem size we compared results across several fixed problem sizes. We ran the Lemke-Howson algorithm on games with 2 players, 150 actions and 2 players, 300 actions. Because Govindan-Wilson is very similar to Lemke-Howson on two-player games and is not optimized for this case [1], we did not run it on these games. We ran Govindan-Wilson and Simplicial Subdivision on games with 6 players, 5 actions and 18 players, 2 actions. For each problem size and distribution, we generated 100 games.

Both to keep our machine-time demands manageable and to keep the graphs in this paper from getting too cluttered, we chose not to use *all* of the GAMUT generators. Instead, we chose a representative slate of 22 distributions from GAMUT. Some of our generators (e.g., Graphical Games, Polymatrix games, and Local Effect Games–LEGs) are parameterized by graph structure; we split these into several sub-distributions based on the kind of graph used.

3 Note that this is a strict subset of graphical games, where payoffs for each player also depend only on the actions of its neighbors, but it is not assumed that payoffs have the additive decomposition.

Suffixes “-CG”, “-RG”, “-SG”, “-SW” and “-Road” indicate, respectively, complete, random, star-shaped, small-world, and road-shaped (see [20]) graphs. Another distribution that we decided to split was the Covariant Game distribution, which implements the random game model of [16]. In this distribution, payoffs for each outcome are generated from a multivariate normal distribution, with correlation between all pairs of players held at some constant ρ. With $\rho = 1$ these games are common-payoff, while $\rho = \frac{-1}{n-1}$ yields minimum correlation and leads to zero-sum games in the two-player case. Rinott and Scarsini show that the probability of the existence of a pure strategy Nash equilibrium in these games varies as a monotonic function of ρ, which makes the games computationally interesting. For these games, suffixes “-Pos”, “-Zero”, and “-Rand” indicate whether ρ was held at 0.9, 0, or drawn uniformly at random from $[\frac{-1}{n-1}, 1]$.

Lemke-Howson, Simplicial Subdivision and Govindan-Wilson are all very complicated path-following numerical algorithms that offer virtually no theoretical guarantees. They all have worst-case running times that are at least exponential, but it is not known whether this bound is tight. On the empirical side, very little previous work has attempted to evaluate these algorithms. The best-known empirical results [11, 21] were obtained for generic games with payoffs drawn independently uniformly at random (in GAMUT, this would be the `RandomGame` generator). Our work may therefore represent the first systematic attempt to understand the empirical behavior of these algorithms on non-generic games.

3.1.2. Experimental Results Figure 2 shows each algorithm’s performance across distributions for two different input sizes. The Y-axis shows CPU time measured in seconds and plotted on a log scale. Column height indicates median runtime over 100 instances, with the error bars showing the 25th and 75th percentiles. The most important thing to note about this graph is that each algorithm exhibits highly variable behavior across our distributions. This is less visible for the Govindan-Wilson algorithm on 18-player games, only because this algorithm’s runtime exceeds our cap for a majority of the problems. However, even on this dataset the error bars demonstrate that the distribution of runtimes varies substantially with the distribution. Moreover, for all three algorithms, we observe that this variation is not an artifact of one particular problem size.

Figure 3 illustrates runtime differences both across and among distributions for 6-player 5-action games. (Though we do not have space to show them here, we observed qualitatively similar results for different input sizes and for the Lemke-Howson algorithm.) Each dot on the graph corresponds to a single run of an algorithm on a game. This graph shows that the distribution of algorithm runtimes varies substantially from one distribution to another, and cannot easily be inferred from 25th/50th/75th quartile figures such as Figure 2. The highly similar Simplicial Subdivision runtimes for Traveler’s Dilemma and Minimum Effort Games are explained by the fact that these games can be solved by iterated elimination of dominated strategies—a step not performed by the GameTracer implementation of Govindan-Wilson. We note that distributions that are related to each other in our taxonomy (e.g., all kinds of Graphical Games, LEGs, or Polymatrix Games) usually give rise to similar—but not identical—algorithmic behavior.

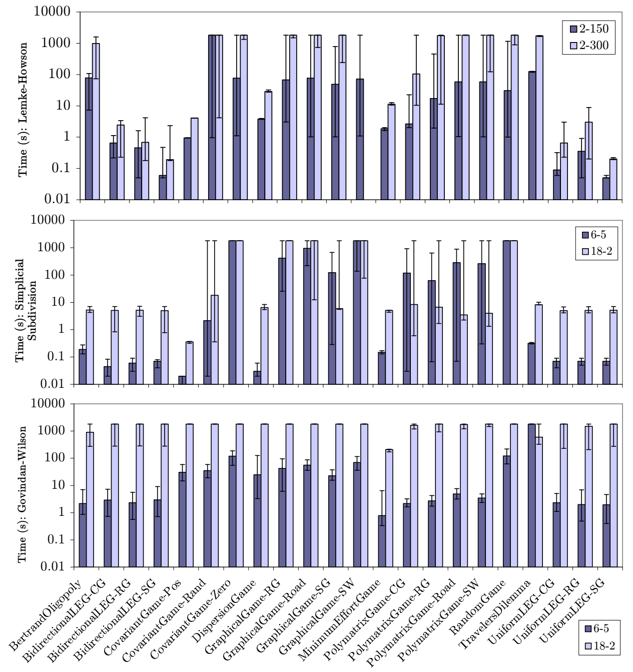


Figure 2. Effect of Problem Size on Solver Performance

Figure 3 makes it clear that algorithms’ runtimes exhibit substantial variation and that algorithms often perform very differently on the same distributions. However, this figure makes it difficult for us to reach conclusions about the extent to which the algorithms are correlated. For an answer to this question, we turn to Figure 4. Each data point represents a single 6-player, 5-action game instance, with the X-axis representing runtime for Simplicial Subdivision and the Y-axis for Govindan-Wilson. Both axes use a log scale. This figure shows that when we focus on instances rather than on distributions, these algorithms are very highly uncorrelated. Simplicial Subdivision does strictly better on 67.2% of the instances, while timing out on 24.7%. Govindan-Wilson wins on 24.7% and times out on 36.5%. It is interesting to note that if a game is easy for Simplicial Subdivision, then it will often be harder for Govindan-Wilson, but in general neither algorithm dominates.

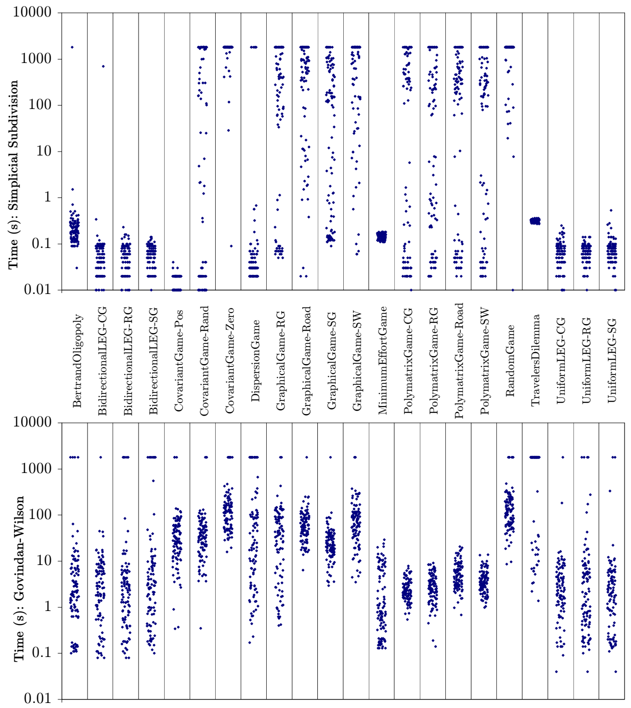


Figure 3. Runtime Distribution for 6-player, 5-action Games

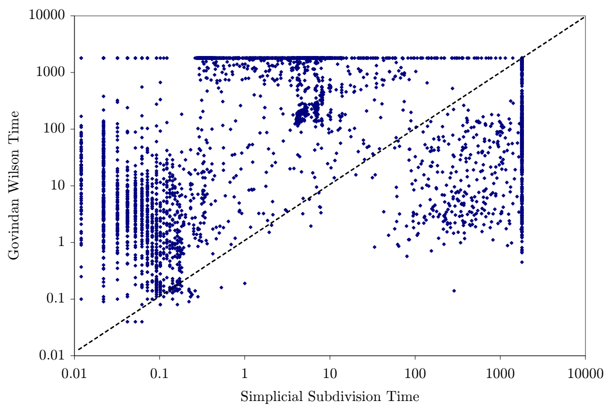


Figure 4. Correlation, 6-player, 5-action.

3.2. Multiagent Learning in Repeated Games

The last few years have seen a surge of research into multiagent learning, resulting in the recent proposal of several new algorithms. This research area is still at a very early stage, particularly with respect to the identification of the best metrics and standards of performance to use for evaluating algorithms. As a result, we do not claim that our results demonstrate anything about the relative merit of the algorithms we study. We believe it is clear, however, that our results show that these algorithms' performance depends crucially on the distributions of games on which they are run, and thus that GAMUT will be a useful tool for researchers in the multiagent learning community.

3.2.1. Experimental Setup We used three learning algorithms: Minimax-Q [10], WoLF [2], and a version of the original Q-learning algorithm for single agent games [22] modified for use by an individual player in a repeated game setting. These algorithms have received much study in recent years; they each have very different performance guarantees, strengths and weaknesses. Single-agent Q-learning assumes away the multiagent component, and thus is not guaranteed to converge at all against an adaptive opponent. Minimax-Q plays a safety-level strategy, and so does not necessarily converge to a best response. WoLF is a variable-learning-rate policy-hill-climbing algorithm that *is* designed to converge to a best response. Previous work in the literature has established that each of these algorithms is very sensitive to its parameter settings (e.g., learning rate) and that the best parameter settings usually vary from one game to the next. Since it is infeasible to perform per-game parameter tuning in an experiment involving tens of thousands of games, we determined parameter values that reproduced previously-published results from [10, 2, 22] and then fixed these parameters for all experiments.

In our experiments we chose to focus on a set of 13 distributions. As before, we keep game sizes constant, this time at 2 actions and 2 players for each game. Although it would also be interesting to study performance in larger games, we decided to focus on a simpler setting in which it would be easier to understand the results of our experiments. For each distribution we generated 100 game instances. For each instance we performed nine different pairings (each possible pairing of the three algorithms, including self-pairings, and in the case of non-self-pairings also allowing each algorithm to play once as player 1 and once as player 2). We ran the algorithms on each pairing ten times, since we found that algorithm performance varied based on the outcomes of coin flips. On each run, we repeated the game 100,000 times. The first 90,000 rounds allow the algorithms to settle into their long-run behavior; we then compute each algorithm's payoff for each game as its average payoff over the following 10,000 rounds. We did this to approximate the offline performance of the learned policy and to minimize the effect of relative differences in the algorithms' learning rates.

3.2.2. Experimental Results There are numerous ways in which learning algorithms can be evaluated. In this section we focus on just two of them. A more comprehensive set of experiments would be required to judge the relative merits of algorithms, but this smaller set of experiments is sufficient to substantiate our claim that algorithm performance varies significantly from one distribution to another.

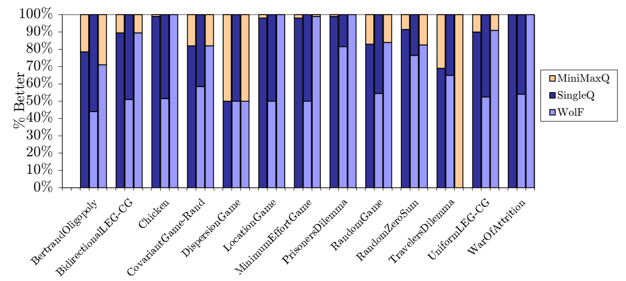


Figure 5. Pairwise Comparison of Algorithms

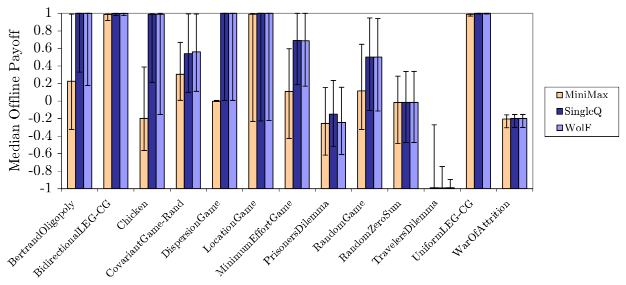


Figure 6. Median Payoffs as Player 1

Figure 5 compares the pairwise performance of three algorithms. The height of a bar along the Y-axis indicates the (normalized) fraction of games in which the corresponding algorithm received a weakly greater payoff than its opponent. In this metric we ignore the magnitude of payoffs, since in general they are incomparable across games. The overall conclusion that we can draw from this graph is that there is great variation in the relative performance of algorithms across distributions. There is no clear "winner"; even Minimax-Q, which is usually outperformed by WoLF, manages to win a significant fraction of games across many distributions, and *dominates* it on Traveler's Dilemma. WoLF and single-agent Q come within 10% of a tie most of the time—suggesting that these algorithms often converge to the same equilibria—but their performance is still far from consistent across different distributions.

Figure 6 compares algorithms using a different metric. Here the Y-axis indicates the average payoff for an algorithm when playing as player 1, with column heights indicating the median and error bars indicating 25th and 75th percentiles. Payoffs are normalized to fall on the range $[-1, 1]$. Despite this normalization, it is difficult to make meaningful comparisons of payoff values across distributions. This graph is interesting because, while focusing on relative performance rather than trying to identify a "winning" algorithm, it demonstrates again that the algorithms' performance varies substantially along the GAMUT. Moreover, this metric shows Minimax-Q to be much more competitive than was suggested by Figure 5.

4. Conclusion

In this paper we presented GAMUT, a game theory test suite. We surveyed hundreds of books and papers to compile a comprehensive database of structured non-generic games and the relationships between them. We built a highly modular and extensible software framework, and used it to implement generators for these sets of games. Finally, we demonstrated the importance of comprehensive test data to game-theoretic algorithms by showing how performance depends crucially on the distribution of instances on which an algorithm is run. We hope that GAMUT will become a useful tool for researchers working at the intersection of game theory and computer science.

Acknowledgments

We would like to thank: Bob Wilson for identifying many useful references and for sharing his insights into the space of games; Rob Powers for providing us with implementations of multiagent learning algorithms; members of Stanford's Multiagent group for many useful comments.

Appendix: GAMUT Implementation

The GAMUT software was built using an object-oriented framework and implemented in Java[4]. Our framework consists of objects in four basic categories: game generators, graphs, functions, and representations. Our main design objective was to make it as easy as possible for end users to write new objects of any of the four kinds, in order to allow GAMUT to be extended to support new sets of games and representations.

Currently, GAMUT contains 35 implementations of `Game` objects, which correspond the 35 procedures we identified in 2.2. They are listed in Table 1. While the internal representations and algorithms used vary depending on the set of games being generated, all of them must be able to return the number of players, the number of actions for each player, and the payoff for a each player for any action profile. `Outputter` classes then encode generated games into appropriate representations.

Many of our generators depend on random graphs (e.g., Graphical Games, Local Effect Games, Polymatrix Games) and functions (e.g., Arms Race, Bertrand Oligopoly, Congestion Games). `Graph` and `Function` classes, listed in Table 2, have been implemented to meet these needs in a

4 See `http://gamut.stanford.edu` for detailed software documentation.

GAMUT Graph Classes:

Barabasi-Albert PLOD	Power-Law Out-Degree
Complete Graph	Random Graph
N-Ary Tree	Ring Graph
N-Dimensional Grid	Small World Graph
N-Dimensional Wrapped Grid	Star Graph

GAMUT Function Classes:

Exponential Function	Polynomial Function
Log Function	Table Function
Decreasing Wrapper	Increasing Polynomial

GAMUT Outputter Classes:

Complete Representations	Incomplete Representations
Default GAMUT Payoff List	Local-Effect Form
Extensive Form	Two-Player Readable Matrix Form
Gambit Normal Form	
Game Tracer Normal Form	
Graphical Form	

Table 2. GAMUT Support Classes

modular way. As with games, additional classes of functions and graphs can be easily added.

`Outputter` classes encapsulate the notion of representation. GAMUT allows for representations to be incomplete and to work only with compatible generators; however, most output representations work with all game generators. Table 2 lists the complete and incomplete representations that are currently supported by GAMUT.

In keeping with our main goal of easy extensibility, GAMUT also implements a wide range of support classes that encapsulate common tasks. For example, GAMUT uses a powerful parameter handling mechanism. Users who want to create a new generator can specify types, ranges, default values and help strings for parameters. Given this information, user help, parsing, and even randomization will all be handled automatically. Since a large (and mundane) part of the user's job now becomes declarative, it is easy to focus on the more interesting and conceptual task of implementing the actual generative algorithm.

Other support utilities offer the ability to convert games into fixed-point arithmetic and to normalize payoffs. The former, besides often being more efficient, sometimes makes more sense game-theoretically: the notion of a Nash equilibrium can become muddy with floating point, since imprecision can lead to equilibrium instability. As mentioned in section 2.2, games' strategic properties are preserved under positive affine transformations. Normalization allows payoff magnitudes to be compared and can avoid machine precision problems.

References

[1] B. Blum, C. Shelton, and D. Koller. A continuation method for Nash equilibria in structured games. In *IJCAI*, 2003.

[2] M. Bowling and M. Veloso. Rational and convergent learning in stochastic games. In *IJCAI*, 2001.

[3] V. Conitzer and T. Sandholm. Complexity results about Nash equilibria. In *IJCAI*, 2003.

[4] I. Gilboa and E. Zemel. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior*, 1, 1989.

[5] S. Govindan and R. Wilson. A global Newton method to compute Nash equilibria. In *J. of Economic Theory*, 2003.

[6] T. Grenager, R. Powers, and Y. Shoham. Dispersion games. In *AAAI*, 2002.

[7] E. Kohlberg and J.F. Mertens. On the strategic stability of equilibria. *Econometrica*, 54, 1986.

[8] C. Lemke and J. Howson. Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, 12, 1964.

[9] K. Leyton-Brown, E. Nudelman, and Y. Shoham. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In *CP*, 2002.

[10] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *ICML*, 1994.

[11] R. McKelvey and A. McLennan. Computation of equilibria in finite games. In H. Amman and D. Kendrick an J. Rust, editors, *Handbook of Computational Economics*, volume I. Elsevier, 1996.

[12] R. D. McKelvey, A. McLennan, and T. Turocy. Gambit: game theory analysis software and tools, 1992. http://econweb.tamu.edu/gambit.

[13] P. Milgrom and J. Roberts. Rationalizability, learning and equilibrium in games with strategic complementarities. *Econometrica*, 58, 1990.

[14] C. Papadimitriou. Algorithms, games, and the internet. In *STOC*, 2001.

[15] A. Rapoport, M. Guyer, and D. Gordon. *The 2x2 Game*. Univeristy of Michigan Press, 1976.

[16] Y. Rinott and M. Scarsini. On the number of pure strategy Nash equilibria in random games. *Games and Economic Behavior*, 33, 2000.

[17] R.W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International J. of Game Theory*, 2, 1973.

[18] W. H. Ruckle. *Geometric Games and Their Applications*. Pitman, 1983.

[19] G. van der Laan, A. Talman, and L. van der Heyden. Simplicial variable dimension algorithms for solving the nonlinear complementarity problem on a product of unit simplices using a general labelling. *Mathematics of Operations Research*, 1987.

[20] D. Vickrey and D. Koller. Multi-agent algorithms for solving graphical games. In *AAAI*, 2002.

[21] B. von Stengel. Computing equilibria for two-person games. In R. Aumann and S. Hart, editors, *Handbook of Game Theory*, volume 3, chapter 45. North-Holland, 2002.

[22] C. J. C. H. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8(3/4), May 1992.

Marginal Contribution Nets: A Compact Representation Scheme for Coalitional Games[*]

Samuel Ieong[†]
Computer Science Department
Stanford University
Stanford, CA 94305
sieong@stanford.edu

Yoav Shoham
Computer Science Department
Stanford University
Stanford, CA 94305
shoham@stanford.edu

ABSTRACT

We present a new approach to representing coalitional games based on rules that describe the marginal contributions of the agents. This representation scheme captures characteristics of the interactions among the agents in a natural and concise manner. We also develop efficient algorithms for two of the most important solution concepts, the Shapley value and the core, under this representation. The Shapley value can be computed in time linear in the size of the input. The emptiness of the core can be determined in time exponential only in the treewidth of a graphical interpretation of our representation.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; J.4 [**Social and Behavioral Sciences**]: Economics; F.2 [**Analysis of Algorithms and Problem Complexity**]

General Terms

Algorithms, Economics

Keywords

Coalitional game theory, Representation, Treewidth

1. INTRODUCTION

Agents can often benefit by coordinating their actions. Coalitional games capture these opportunities of coordination by explicitly modeling the ability of the agents to take joint actions as primitives. As an abstraction, coalitional games assign a payoff to each group of agents in the game. This payoff is intended to reflect the payoff the group of agents can secure for themselves regardless of the actions of the agents not in the group. These choices of primitives are in contrast to those of non-cooperative games, of which agents are modeled independently, and their payoffs depend critically on the actions chosen by the other agents.

1.1 Coalitional Games and E-Commerce

Coalitional games have appeared in the context of e-commerce. In [7], Kleinberg *et al.* use coalitional games to study recommendation systems. In their model, each individual knows about a certain set of items, is interested in learning about all items, and benefits from finding out about them. The payoffs to groups of agents are the total number of distinct items known by its members. Given this coalitional game setting, Kleinberg *et al.* compute the value of the private information of the agents is worth to the system using the solution concept of the Shapley value (definition can be found in section 2). These values can then be used to determine how much each agent should receive for participating in the system.

As another example, consider the economics behind supply chain formation. The increased use of the Internet as a medium for conducting business has decreased the costs for companies to coordinate their actions, and therefore coalitional game is a good model for studying the supply chain problem. Suppose that each manufacturer purchases his raw materials from some set of suppliers, and that the suppliers offer higher discount with more purchases. The decrease in communication costs will let manufacturers find others interested in the same set of suppliers cheaper, and facilitates formation of coalitions to bargain with the suppliers. Depending on the set of suppliers and how much from each supplier each coalition purchases, we can assign payoffs to the coalitions depending on the discount it receives. The resulting game can be analyzed using coalitional game theory, and we can answer questions such as the stability of coalitions, and how to fairly divide the benefits among the participating manufacturers. A similar problem, combinatorial coalition formation, has previously been studied in [8].

1.2 Evaluation Criteria for Coalitional Game Representation

To capture the coalitional games described above and perform computations on them, we must first find a representation for these games. The naïve solution is to enumerate

[*]This research is supported by DARPA grant F30602-00-2-0598 and NSF grant ITR0205633.

[†]Samuel Ieong is supported by a Richard and Naomi Horowitz Stanford Graduate Fellowship.

the payoffs to each set of agents, therefore requiring space exponential in the number of agents in the game. For the two applications described, the number of agents in the system can easily exceed a hundred; this naïve approach will not be scalable to such problems. Therefore, it is critical to find good representation schemes for coalitional games.

We believe that the quality of a representation scheme should be evaluated by four criteria.

Expressivity: the breadth of the class of coalitional games covered by the representation.

Conciseness: the space requirement of the representation.

Efficiency: the efficiency of the algorithms we can develop for the representation.

Simplicity: the ease of use of the representation by users of the system.

The ideal representation should be fully expressive, i.e., it should be able to represent any coalitional games, use as little space as possible, have efficient algorithms for computation, and be easy to use. The goal of this paper is to develop a representation scheme that has properties close to the ideal representation.

Unfortunately, given that the number of degrees of freedom of coalitional games is $O(2^n)$, not all games can be represented concisely using a single scheme due to information theoretic constraints. For any given class of games, one may be able to develop a representation scheme that is tailored and more compact than a general scheme. For example, for the recommendation system game, a highly compact representation would be one that simply states which agents know of which products, and let the algorithms that operate on the representation to compute the values of coalitions appropriately. For some problems, however, there may not be efficient algorithms for customized representations. By having a general representation and efficient algorithms that go with it, the representation will be useful as a prototyping tool for studying new economic situations.

1.3 Previous Work

The question of coalitional game representation has only been sparsely explored in the past [2, 3, 4]. In [4], Deng and Papadimitriou focused on the complexity of different solution concepts on coalitional games defined on graphs. While the representation is compact, it is not fully expressive. In [2], Conitzer and Sandholm looked into the problem of determining the emptiness of the core in superadditive games. They developed a compact representation scheme for such games, but again the representation is not fully expressive either. In [3], Conitzer and Sandholm developed a fully expressive representation scheme based on decomposition. Our work extends and generalizes the representation schemes in [3, 4] through decomposing the game into a set of rules that assign marginal contributions to groups of agents. We will give a more detailed review of these papers in section 2.2 after covering the technical background.

1.4 Summary of Our Contributions

- We develop the *marginal contribution networks* representation, a fully expressive representation scheme whose size scales according to the complexity of the interactions among the agents. We believe that the representation is also simple and intuitive.
- We develop an algorithm for computing the Shapley value of coalitional games under this representation that runs in time linear in the size of the input.
- Under the graphical interpretation of the representation, we develop an algorithm for determining the whether a payoff vector is in the core and the emptiness of the core in time exponential only in the treewidth of the graph.

2. PRELIMINARIES

In this section, we will briefly review the basics of coalitional game theory and its two primary solution concepts, the Shapley value and the core.[1] We will also review previous work on coalitional game representation in more detail. Throughout this paper, we will assume that the payoff to a group of agents can be freely distributed among its members. This assumption is often known as the *transferable utility* assumption.

2.1 Technical Background

We can represent a coalition game with transferable utility by the pair $\langle N, v\rangle$, where

- N is the set of agents; and
- $v : 2^N \mapsto \mathbb{R}$ is a function that maps each group of agents $S \subseteq N$ to a real-valued payoff.

This representation is known as the characteristic form. As there are exponentially many subsets, it will take space exponential in the number of agents to describe a coalitional game.

An *outcome* in a coalitional game specifies the utilities the agents receive. A *solution concept* assigns to each coalitional game a set of "reasonable" outcomes. Different solution concepts attempt to capture in some way outcomes that are stable and/or fair. Two of the best known solution concepts are the Shapley value and the core.

The Shapley value is a normative solution concept. It prescribes a "fair" way to divide the gains from cooperation when the grand coalition (i.e., N) is formed. The division of payoff to agent i is the average marginal contribution of agent i over all possible permutations of the agents. Formally, let $\phi_i(v)$ denote the Shapley value of i under characteristic function v, then[2]

$$\phi_i(v) = \sum_{S \subset N} \frac{s!(n-s-1)!}{n!} \left(v(S \cup \{i\}) - v(S)\right) \qquad (1)$$

The Shapley value is a solution concept that satisfies many nice properties, and has been studied extensively in the economic and game theoretic literature. It has a very useful axiomatic characterization.

Efficiency (EFF) A total of $v(N)$ is distributed to the agents, i.e., $\sum_{i \in N} \phi_i(v) = v(N)$.

[1]The materials and terminology are based on the textbooks by Mas-Colell *et al.* [9] and Osborne and Rubinstein [11].

[2]As a notational convenience, we will use the lower-case letter to represent the cardinality of a set denoted by the corresponding upper-case letter.

Symmetry (SYM) If agents i and j are interchangeable, then $\phi_i(v) = \phi_j(v)$.

Dummy (DUM) If agent i is a dummy player, i.e., his marginal contribution to all groups S are the same, $\phi_i(v) = v(\{i\})$.

Additivity (ADD) For any two coalitional games v and w defined over the same set of agents N, $\phi_i(v + w) = \phi_i(v) + \phi_i(w)$ for all $i \in N$, where the game $v + w$ is defined as $(v + w)(S) = v(S) + w(S)$ for all $S \subseteq N$.

We will refer to these axioms later in our proof of correctness of the algorithm for computing the Shapley value under our representation in section 4.

The core is another major solution concept for coalitional games. It is a descriptive solution concept that focuses on outcomes that are "stable." Stability under core means that no set of players can jointly deviate to improve their payoffs. Formally, let $x(S)$ denote $\sum_{i \in S} x_i$. An outcome $x \in \mathbb{R}^n$ is in the core if

$$\forall S \subseteq N \quad x(S) \geq v(S) \tag{2}$$

The core was one of the first proposed solution concepts for coalitional games, and had been studied in detail. An important question for a given coalitional game is whether the core is empty. In other words, whether there is any outcome that is stable relative to group deviation. For a game to have a non-empty core, it must satisfy the property of *balancedness*, defined as follows. Let $1_S \in \mathbb{R}^n$ denote the characteristic vector of S given by

$$(1_S)_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases}$$

Let $(\lambda_S)_{S \subseteq N}$ be a set of weights such that each λ_S is in the range between 0 and 1. This set of weights, $(\lambda_S)_{S \subseteq N}$, is a *balanced collection* if for all $i \in N$,

$$\sum_{S \subseteq N} \lambda_S (1_S)_i = 1$$

A game is *balanced* if for all balanced collections of weights,

$$\sum_{S \subseteq N} \lambda_S v(S) \leq v(N) \tag{3}$$

By the Bondereva-Shapley theorem, the core of a coalitional game is non-empty if and only if the game is balanced. Therefore, we can use linear programming to determine whether the core of a game is empty.

$$\begin{array}{lll} \underset{\lambda \in \mathbb{R}^{2^n}}{\text{maximize}} & \sum_{S \subseteq N} \lambda_S v(S) & \\ \text{subject to} & \sum_{S \subseteq N} \lambda_S 1_S = 1 & \forall i \in N \\ & \lambda_S \geq 0 & \forall S \subseteq N \end{array} \tag{4}$$

If the optimal value of (4) is greater than the value of the grand coalition, then the core is empty. Unfortunately, this program has an exponential number of variables in the number of players in the game, and hence an algorithm that operates directly on this program would be infeasible in practice. In section 5.4, we will describe an algorithm that answers the question of emptiness of core that works on the dual of this program instead.

2.2 Previous Work Revisited

Deng and Papadimitriou looked into the complexity of various solution concepts on coalitional games played on weighted graphs in [4]. In their representation, the set of agents are the nodes of the graph, and the value of a set of agents S is the sum of the weights of the edges spanned by them. Notice that this representation is concise since the space required to specify such a game is $O(n^2)$. However, this representation is not general; it will not be able to represent interactions among three or more agents. For example, it will not be able to represent the *majority game*, where a group of agents S will have value of 1 if and only if $s > n/2$. On the other hand, there is an efficient algorithm for computing the Shapley value of the game, and for determining whether the core is empty under the restriction of positive edge weights. However, in the unrestricted case, determining whether the core is non-empty is coNP-complete.

Conitzer and Sandholm in [2] considered coalitional games that are superadditive. They described a concise representation scheme that only states the value of a coalition if the value is *strictly* superadditive. More precisely, the semantics of the representation is that for a group of agents S,

$$v(S) = \max_{\{T_1, T_2, \ldots, T_n\} \in \Pi} \sum_i v(T_i)$$

where Π is the set of all possible partitions of S. The value $v(S)$ is only explicitly specified for S if $v(S)$ is greater than all partitioning of S other than the trivial partition ($\{S\}$). While this representation can represent all games that are superadditive, there are coalitional games that it cannot represent. For example, it will not be able to represent any games with substitutability among the agents. An example of a game that cannot be represented is the *unit game*, where $v(S) = 1$ as long as $S \neq \emptyset$. Under this representation, the authors showed that determining whether the core is non-empty is coNP-complete. In fact, even determining the value of a group of agents is NP-complete.

In a more recent paper, Conitzer and Sandholm described a representation that decomposes a coalitional game into a number of subgames whose sum add up to the original game [3]. The payoffs in these subgames are then represented by their respective characteristic functions. This scheme is fully general as the characteristic form is a special case of this representation. For any given game, there may be multiple ways to decompose the game, and the decomposition may influence the computational complexity. For computing the Shapley value, the authors showed that the complexity is linear in the input description; in particular, if the largest subgame (as measured by number of agents) is of size n and the number of subgames is m, then their algorithm runs in $O(m2^n)$ time, where the input size will also be $O(m2^n)$. On the other hand, the problem of determining whether a certain outcome is in the core is coNP-complete.

3. MARGINAL CONTRIBUTION NETS

In this section, we will describe the *Marginal Contribution Networks* representation scheme. We will show that the idea is flexible, and we can easily extend it to increase its conciseness. We will also show how we can use this scheme to represent the recommendation game from the introduction. Finally, we will show that this scheme is fully expressive, and generalizes the representation schemes in [3, 4].

3.1 Rules and Marginal Contribution Networks

The basic idea behind marginal contribution networks (MC-nets) is to represent coalitional games using sets of *rules*. The rules in MC-nets have the following syntactic form:

$$\textit{Pattern} \ \rightarrow \textit{value}$$

A rule is said to *apply* to a group of agents S if S *meets* the requirement of the *Pattern*. In the basic scheme, these patterns are conjunctions of agents, and S meets the requirement of the given pattern if S is a superset of it. The value of a group of agents is defined to be the sum over the values of all rules that apply to the group. For example, if the set of rules are

$$\{a \wedge b\} \rightarrow 5$$
$$\{b\} \rightarrow 2$$

then $v(\{a\}) = 0$, $v(\{b\}) = 2$, and $v(\{a, b\}) = 5 + 2 = 7$.

MC-nets is a very flexible representation scheme, and can be extended in different ways. One simple way to extend it and increase its conciseness is to allow a wider class of patterns in the rules. A pattern that we will use throughout the remainder of the paper is one that applies only in the *absence* of certain agents. This is useful for expressing concepts such as substitutability or default values. Formally, we express such patterns by

$$\{p_1 \wedge p_2 \wedge \ldots \wedge p_m \wedge \neg n_1 \wedge \neg n_2 \wedge \ldots \wedge \neg n_n\}$$

which has the semantics that such rule will apply to a group S only if $\{p_i\}_{i=1}^m \in S$ and $\{n_j\}_{j=1}^n \notin S$. We will call the $\{p_i\}_{i=1}^m$ in the above pattern the *positive literals*, and $\{n_j\}_{j=1}^n$ the *negative literals.* Note that if the pattern of a rule consists solely of negative literals, we will consider that the empty set of agents will also satisfy such pattern, and hence $v(\emptyset)$ may be non-zero in the presence of negative literals.

To demonstrate the increase in conciseness of representation, consider the unit game described in section 2.2. To represent such a game without using negative literals, we will need 2^n rules for n players: we need a rule of value 1 for each individual agent, a rule of value -1 for each pair of agents to counter the double-counting, a rule of value 1 for each triplet of agents, etc., similar to the inclusion-exclusion principle. On the other hand, using negative literals, we only need n rules: value 1 for the first agent, value 1 for the second agent *in the absence of the first agent*, value 1 for the third agent *in the absence of the first two agents*, etc. The representational savings can be exponential in the number of agents.

Given a game represented as a MC-net, we can interpret the set of rules that make up the game as a graph. We call this graph the *agent graph.* The nodes in the graph will represent the agents in the game, and for each rule in the MC-net, we connect all the agents in the rule together and assign a value to the *clique* formed by the set of agents. Notice that to accommodate negative literals, we will need to annotate the clique appropriately. This alternative view of MC-nets will be useful in our algorithm for CORE-MEMBERSHIP in section 5.

We would like to end our discussion of the representation scheme by mentioning a trade-off between the expressiveness of patterns and the space required to represent them. To represent a coalitional game in characteristic form, one would need to specify all $2^n - 1$ values. There is no overhead on top of that since there is a natural ordering of the groups. For MC-nets, however, specification of the rules requires specifying both the patterns and the values. The patterns, if not represented compactly, may end up overwhelming the savings from having fewer values to specify. The space required for the patterns also leads to a trade-off between the expressiveness of the allowed patterns and the simplicity of representing them. However, we believe that for most naturally arising games, there should be sufficient structure in the problem such that our representation achieves a net saving over the characteristic form.

3.2 Example: Recommendation Game

As an example, we will use MC-net to represent the recommendation game discussed in the introduction. For each product, as the benefit of knowing about the product will count only once for each group, we need to capture substitutability among the agents. This can be captured by a scaled unit game. Suppose the value of the knowledge about product i is v_i, and there are n_i agents, denoted by $\{x_i^j\}$, who know about the product, the game for product i can then be represented as the following rules:

$$\{x_i^1\} \rightarrow v_i$$
$$\{x_i^2 \wedge \neg x_i^1\} \rightarrow v_i$$
$$\vdots$$
$$\{x_i^{n_i} \wedge \neg x_i^{n_i - 1} \wedge \cdots \wedge \neg x_i^1\} \rightarrow v_i$$

The entire game can then be built up from the sets of rules of each product. The space requirement will be $O(mn^*)$, where m is the number of products in the system, and n^* is the maximum number of agents who knows of the same product.

3.3 Representation Power

We will discuss the expressiveness and conciseness of our representation scheme and compare it with the previous works in this subsection.

PROPOSITION 1. *Marginal contribution networks constitute a fully expressive representation scheme.*

PROOF. Consider an arbitrary coalitional game $\langle N, v\rangle$ in characteristic form representation. We can construct a set of rules to describe this game by starting from the singleton sets and building up the set of rules. For any singleton set $\{i\}$, we create a rule $\{i\} \rightarrow v(i)$. For any pair of agents $\{i, j\}$, we create a rule $\{i \wedge j\} \rightarrow v(\{i, j\}) - v(\{i\}) - v(\{j\}$. We can continue to build up rules in a manner similar to the inclusion-exclusion principle. Since the game is arbitrary, MC-nets are fully expressive. □

Using the construction outlined in the proof, we can show that our representation scheme can simulate the multi-issue representation scheme of [3] in almost the same amount of space.

PROPOSITION 2. *Marginal contribution networks use at most a linear factor (in the number of agents) more space than multi-issue representation for any game.*

PROOF. Given a game in multi-issue representation, we start by describing each of the subgames, which are represented in characteristic form in [3], with a set of rules. We then build up the grand game by including all the rules from the subgames. Note that our representation may require a space larger by a linear factor due to the need to describe the patterns for each rule. On the other hand, our approach may have fewer than exponential number of rules for each subgame, depending on the structure of these subgames, and therefore may be more concise than multi-issue representation. □

On the other hand, there are games that require exponentially more space to represent under the multi-issue scheme compared to our scheme.

PROPOSITION 3. *Marginal contribution networks are exponentially more concise than multi-issue representation for certain games.*

PROOF. Consider a unit game over all the agents N. As explained in 3.1, this game can be represented in linear space using MC-nets with negative literals. However, as there is no decomposition of this game into smaller subgames, it will require space $O(2^n)$ to represent this game under the multi-issue representation. □

Under the agent graph interpretation of MC-nets, we can see that MC-nets is a generalization of the graphical representation in [4], namely from weighted graphs to weighted hypergraphs.

PROPOSITION 4. *Marginal contribution networks can represent any games in graphical form (under [4]) in the same amount of space.*

PROOF. Given a game in graphical form, G, for each edge (i, j) with weight w_{ij} in the graph, we create a rule $\{i, j\} \to w_{ij}$. Clearly this takes exactly the same space as the size of G, and by the additive semantics of the rules, it represents the same game as G. □

4. COMPUTING THE SHAPLEY VALUE

Given a MC-net, we have a simple algorithm to compute the Shapley value of the game. Considering each rule as a separate game, we start by computing the Shapley value of the agents for each rule. For each agent, we then sum up the Shapley values of that agent over all the rules. We first show that this final summing process correctly computes the Shapley value of the agents.

PROPOSITION 5. *The Shapley value of an agent in a marginal contribution network is equal to the sum of the Shapley values of that agent over each rule.*

PROOF. For any group S, under the MC-nets representation, $v(S)$ is defined to be the sum over the values of all the rules that apply to S. Therefore, considering each rule as a game, by the (ADD) axiom discussed in section 2, the Shapley value of the game created from aggregating all the rules is equal to the sum of the Shapley values over the rules. □

The remaining question is how to compute the Shapley values of the rules. We can separate the analysis into two cases, one for rules with only positive literals and one for rules with mixed literals.

For rules that have only positive literals, the Shapley value of the agents is v/m, where v is the value of the rule and m is the number of agents in the rule. This is a direct consequence of the (SYM) axiom of the Shapley value, as the agents in a rule are indistinguishable from each other.

For rules that have both positive and negative literals, we can consider the positive and the negative literals separately. For a given positive literal i, the rule will apply only if i occurs in a given permutation after the rest of the positive literals but before any of the negative literals. Formally, let ϕ_i denote the Shapley value of i, p denote the cardinality of the positive set, and n denote the cardinality of the negative set, then

$$\phi_i = \frac{(p-1)!n!}{(p+n)!}\, v = \frac{v}{p\binom{p+n}{n}}$$

For a given negative literal j, j will be responsible for cancelling the application of the rule if all positive literals come before the negative literals in the ordering, and j is the first among the negative literals. Therefore,

$$\phi_j = \frac{p!(n-1)!}{(p+n)!}\,(-v) = \frac{-v}{n\binom{p+n}{p}}$$

By the (SYM) axiom, all positive literals will have the value of ϕ_i and all negative literals will have the value of ϕ_j.

Note that the sum over all agents in rules with mixed literals is 0. This is to be expected as these rules contribute 0 to the grand coalition. The fact that these rules have no effect on the grand coalition may appear odd at first. But this is because the presence of such rules is to define the values of coalitions smaller than the grand coalition.

In terms of computational complexity, given that the Shapley value of any agent in a given rule can be computed in time linear in the pattern of the rule, the total running time of the algorithm for computing the Shapley value of the game is linear in the size of the input.

5. ANSWERING CORE-RELATED QUESTIONS

There are a few different but related computational problems associated with the solution concept of the core. We will focus on the following two problems:

Definition 1. (CORE-MEMBERSHIP) Given a coalitional game and a payoff vector x, determine if x is in the core.

Definition 2. (CORE-NON-EMPTINESS) Given a coalitional game, determine if the core is non-empty.

In the rest of the section, we will first show that these two problems are coNP-complete and coNP-hard respectively, and discuss some complexity considerations about these problems. We will then review the main ideas of tree decomposition as it will be used extensively in our algorithm for CORE-MEMBERSHIP. Next, we will present the algorithm for CORE-MEMBERSHIP, and show that the algorithm runs in polynomial time for graphs of bounded treewidth. We end by extending this algorithm to answer the question of CORE-NON-EMPTINESS in polynomial time for graphs of bounded treewidth.

5.1 Computational Complexity

The hardness of Core-Membership and Core-Non-Emptiness follows directly from the hardness results of games over weighted graphs in [4].

Proposition 6. Core-Membership *for games represented as marginal contribution networks is coNP-complete.*

Proof. Core-Membership in MC-nets is in the class of coNP since any set of agents S of which $v(S) > x(S)$ will serve as a certificate to show that x does not belong to the core. As for its hardness, given any instance of Core-Membership for a game in graphical form of [4], we can encode the game in exactly the same space using MC-net due to Proposition 4. Since Core-Membership for games in graphical form is coNP-complete, Core-Membership in MC-nets is coNP-hard. □

Proposition 7. Core-Non-Emptiness *for games represented as marginal contribution networks is coNP-hard.*

Proof. The same argument for hardness between games in graphical frm and MC-nets holds for the problem of Core-Non-Emptiness. □

We do not know of a certificate to show that Core-Non-Emptiness is in the class of coNP as of now. Note that the "obvious" certificate of a balanced set of weights based on the Bondereva-Shapley theorem is exponential in size. In [4], Deng and Papadimitriou showed the coNP-completeness of Core-Non-Emptiness via a combinatorial characterization, namely that the core is non-empty if and only if there is no negative cut in the graph. In MC-nets, however, there need not be a negative hypercut in the graph for the core to be empty, as demonstrated by the following game ($N = \{1, 2, 3, 4\}$):

$$v(S) = \begin{cases} 1 & \text{if } S = \{1,2,3,4\} \\ 3/4 & \text{if } S = \{1,2\}, \{1,3\}, \{1,4\}, \text{ or } \{2,3,4\} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

Applying the Bondereva-Shapley theorem, if we let $\lambda_{12} = \lambda_{13} = \lambda_{14} = 1/3$, and $\lambda_{234} = 2/3$, this set of weights demonstrates that the game is not balanced, and hence the core is empty. On the other hand, this game can be represented with MC-nets as follows (weights on hyperedges):

$$w(\{1,2\}) = w(\{1,3\}) = w(\{1,4\}) = 3/4$$
$$w(\{1,2,3\}) = w(\{1,2,4\}) = w(\{1,3,4\}) = -6/4$$
$$w(\{2,3,4\}) = 3/4$$
$$w(\{1,2,3,4\}) = 10/4$$

No matter how the set is partitioned, the sum over the weights of the hyperedges in the cut is always non-negative.

To overcome the computational hardness of these problems, we have developed algorithms that are based on tree decomposition techniques. For Core-Membership, our algorithm runs in time exponential only in the treewidth of the agent graph. Thus, for graphs of small treewidth, such as trees, we have a tractable solution to determine if a payoff vector is in the core. By using this procedure as a separation oracle, i.e., a procedure for returning the inequality violated by a candidate solution, to solving a linear program that is related to Core-Non-Emptiness using the ellipsoid method, we can obtain a polynomial time algorithm for Core-Non-Emptiness for graphs of bounded treewidth.

5.2 Review of Tree Decomposition

As our algorithm for Core-Membership relies heavily on tree decomposition, we will first briefly review the main ideas in tree decomposition and treewidth.[3]

Definition 3. A *tree decomposition* of a graph $G = (V, E)$ is a pair $(\mathcal{X}, T)$, where $T = (I, F)$ is a tree and $\mathcal{X} = \{X_i \mid i \in I\}$ is a family of subsets of V, one for each node of T, such that

- $\bigcup_{i \in I} X_i = V$;
- For all edges $(v, w) \in E$, there exists an $i \in I$ with $v \in X_i$ and $w \in X_i$; and
- (*Running Intersection Property*) For all $i, j, k \in I$: if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

The treewidth of a tree decomposition is defined as the maximum cardinality over all sets in $\mathcal{X}$, less one. The treewidth of a graph is defined as the minimum treewidth over all tree decompositions of the graph.

Given a tree decomposition, we can convert it into a *nice* tree decomposition of the same treewidth, and of size linear in that of T.

Definition 4. A tree decomposition T is *nice* if T is rooted and has four types of nodes:

Leaf nodes i are leaves of T with $|X_i| = 1$.

Introduce nodes i have one child j such that $X_i = X_j \cup \{v\}$ of some $v \in V$.

Forget nodes i have one child j such that $X_i = X_j \setminus \{v\}$ for some $v \in X_j$.

Join nodes i have two children j and k with $X_i = X_j = X_k$.

An example of a (partial) nice tree decomposition together with a classification of the different types of nodes is in Figure 1. In the following section, we will refer to nodes in the tree decomposition as nodes, and nodes in the agent graph as agents.

5.3 Algorithm for Core Membership

Our algorithm for Core-Membership takes as an input a nice tree decomposition T of the agent graph and a payoff vector x. By definition, if x belongs to the core, then for all groups $S \subseteq N$, $x(S) \geq v(S)$. Therefore, the difference $x(S) - v(S)$ measures how "close" the group S is to violating the core condition. We call this difference the *excess* of group S.

Definition 5. The *excess* of a coalition S, $e(S)$, is defined as $x(S) - v(S)$.

A brute-force approach to determine if a payoff vector belongs to the core will have to check that the excesses of all groups are non-negative. However, this approach ignores the structure in the agent graph that will allow an algorithm to infer that certain groups have non-negative excesses due to

[3]This is based largely on the materials from a survey paper by Bodlaender [1].

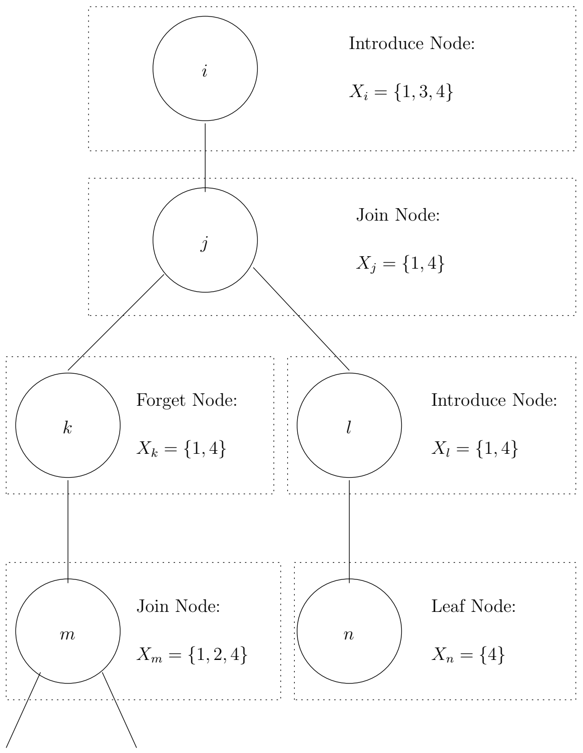


Figure 1: Example of a (partial) nice tree decomposition

the excesses computed elsewhere in the graph. Tree decomposition is the key to take advantage of such inferences in a structured way.

For now, let us focus on rules with positive literals. Suppose we have already checked that the excesses of all sets $R \subseteq U$ are non-negative, and we would like to check if the addition of an agent i to the set U will create a group with negative excess. A naïve solution will be to compute the excesses of all sets that include i. The excess of the group $(R \cup \{i\})$ for any group R can be computed as follows

$$e(R \cup \{i\}) = e(R) + x_i - v(c) \tag{6}$$

where c is the cut between R and i, and $v(c)$ is the sum of the weights of the edges in the cut.

However, suppose that from the tree decomposition, we know that i is only connected to a subset of U, say S, which we will call the *entry set* to U. Ideally, because i does not share any edges with members of $\bar{U} = (U \setminus S)$, we would hope that an algorithm can take advantage of this structure by checking only sets that are subsets of $(S \cup \{i\})$. This computational saving may be possible since $(x_i - v(c))$ in the update equation of (6) does not depend on $\bar{U}$. However, we cannot simply ignore $\bar{U}$ as members of $\bar{U}$ may still influence the excesses of groups that include agent i through group S. Specifically, if there exists a group $T \supset S$ such that $e(T) < e(S)$, then even when $e(S \cup \{i\})$ has non-negative excess, $e(T \cup \{i\})$ may have negative excess. In other words, the excess available at S may have been "drained" away due to T. This motivates the definition of the *reserve* of a group.

Definition 6. The *reserve* of a coalition S relative to a coalition U is the minimum excess over all coalitions between S and U, i.e., all $T : S \subseteq T \subseteq U$. We denote this value by $r(S,U)$. We will refer to the group T that has the minimum excess as $\arg r(S,U)$. We will also call U the *limiting set* of the reserve and S the *base set* of the reserve.

Our algorithm works by keeping track of the reserves of all non-empty subsets that can be formed by the agents of a node at each of the nodes of the tree decomposition. Starting from the leaves of the tree and working towards the root, at each node i, our algorithm computes the reserves of all groups $S \subseteq X_i$, limited by the set of agents in the subtree rooted at i, T_i, *except* those in $(X_i \setminus S)$. The agents in $(X_i \setminus S)$ are excluded to ensure that S is an entry set. Specifically, S is the entry set to $((T_i \setminus X_i) \cup S)$.

To accomodate for negative literals, we will need to make two adjustments. Firstly, the cut between an agent m and a set S at node i now refers to the cut among agent m, set S, and set $\neg(X_i \setminus S)$, and its value must be computed accordingly. Also, when an agent m is introduced to a group at an introduce node, we will also need to consider the change in the reserves of groups that do not include m due to possible cut involving $\neg m$ and the group.

As an example of the reserve values we keep track of at a tree node, consider node i of the tree in Figure 1. At node i, we will keep track of the following:

$$\begin{aligned}
&r(\{1\},\{1,2,\ldots\})\\
&r(\{3\},\{2,3,\ldots\})\\
&r(\{4\},\{2,4,\ldots\})\\
&r(\{1,3\},\{1,2,3,\ldots\})\\
&r(\{1,4\},\{1,2,4,\ldots\})\\
&r(\{3,4\},\{2,3,4,\ldots\})\\
&r(\{1,3,4\},\{1,2,3,4,\ldots\}
\end{aligned}$$

where the dots $\ldots$ refer to the agents rooted under node m.

For notational use, we will use $r_i(S)$ to denote $r(S,U)$ at node i where U is the set of agents in the subtree rooted at node i excluding agents in $(X_i \setminus S)$. We sometimes refer to these values as the *r-values* of a node. The details of the r-value computations are in Algorithm 1.

To determine if the payoff vector x is in the core, during the r-value computation at each node, we can check if all of the r-values are non-negative. If this is so for all nodes in the tree, the payoff vector x is in the core. The correctness of the algorithm is due to the following proposition.

Proposition 8. *The payoff vector x is not in the core if and only if the r-values at some node i for some group S is negative.*

Proof. ($\Leftarrow$) If the reserve at some node i for some group S is negative, then there exists a coalition T for which $e(T) = x(T) - v(T) < 0$, hence x is not in the core.

($\Rightarrow$) Suppose x is not in the core, then there exists some group R^* such that $e(R^*) < 0$. Let X_{root} be the set of nodes at the root. Consider any set $S \in X_{root}$, $r_{root}(S)$ will have the base set of S and the limiting set of $((N \setminus X_{root}) \cup S)$. The union over all of these ranges includes all sets U for which $U \cap X_{root} \neq \emptyset$. Therefore, if R^* is not disjoint from X_{root}, the r-value for some group in the root is negative.

If R^* is disjoint from U, consider the forest $\{T_i\}$ resulting from removal of all tree nodes that include agents in X_{root}.

Algorithm 1 Subprocedures for Core Membership

```
LEAF-NODE($i$)
 1: $r_i(X_i) \leftarrow e(X_i)$
INTRODUCE-NODE($i$)
 2: $j \leftarrow$ child of $i$
 3: $m \leftarrow X_i \setminus X_j$ {the introduced node}
 4: for all $S \subseteq X_j, S \neq \emptyset$ do
 5:     $C \leftarrow$ all hyperedges in the cut of $m$, $S$, and $\neg(X_i \setminus S)$
 6:     $r_i(S \cup \{x\}) \leftarrow r_j(S) + x_m - v(C)$
 7:     $C \leftarrow$ all hyperedges in the cut of $\neg m$, $S$, and $\neg(X_i \setminus S)$
 8:     $r_i(S) \leftarrow r_j(S) - v(C)$
 9: end for
10: $r(\{m\}) \leftarrow e(\{m\})$
FORGET-NODE($i$)
11: $j \leftarrow$ child of $i$
12: $m \leftarrow X_j \setminus X_i$ {the forgotten node}
13: for all $S \subseteq X_i, S \neq \emptyset$ do
14:     $r_i(S) = \min(r_j(S), r_j(S \cup \{m\}))$
15: end for
JOIN-NODE($i$)
16: $\{j, k\} \leftarrow$ {left, right} child of $i$
17: for all $S \subseteq X_i, S \neq \emptyset$ do
18:     $r_i(S) \leftarrow r_j(S) + r_k(S) - e(S)$
19: end for
```

By the running intersection property, the sets of nodes in the trees T_i's are disjoint. Thus, if the set $R^* = \bigcup_i S_i$ for some $S_i \in T_i$, $e(R^*) = \sum_i e(S_i) < 0$ implies some group S_i^* has negative excess as well. Therefore, we only need to check the r-values of the nodes on the individual trees in the forest.

But for each tree in the forest, we can apply the same argument restricted to the agents in the tree. In the base case, we have the leaf nodes of the original tree decomposition, say, for agent i. If $R^* = \{i\}$, then $r(\{i\}) = e(\{i\}) < 0$. Therefore, by induction, if $e(R^*) < 0$, some reserve at some node would be negative. □

We will next explain the intuition behind the correctness of the computations for the r-values in the tree nodes. A detailed proof of correctness of these computations can be found in the appendix under Lemmas 1 and 2.

PROPOSITION 9. *The procedure in Algorithm 1 correctly compute the r-values at each of the tree nodes.*

PROOF. (SKETCH) We can perform a case analysis over the four types of tree nodes in a nice tree decomposition.

Leaf nodes (i) The only reserve value to be computed is $r_i(X_i)$, which equals $r(X_i, X_i)$, and therefore it is just the excess of group X_i.

Forget nodes (i with child j) Let m be the forgotten node. For any subset $S \subseteq X_i$, $\arg r_i(S)$ must be chosen between the groups of S and $S \cup \{m\}$, and hence we choose between the lower of the two from the r-values at node j.

Introduce nodes (i with child j) Let m be the introduced node. For any subset $T \subseteq X_i$ that includes m, let S denote $(T \setminus \{m\})$. By the running intersection property, there are no rules that involve m and agents of the subtree rooted at node i except those involving m and agents in X_i. As both the base set and the limiting set of the r-values of node j and node i differ by $\{m\}$, for any group V that lies between the base set and the limiting set of node i, the excess of group V will differ by a constant amount from the corresponding group $(V \setminus \{m\})$ at node j. Therefore, the set $\arg r_i(T)$ equals the set $\arg r_j(S) \cup \{m\}$, and $r_i(T) = r_j(S) + x_m - v(\text{cut})$, where $v(\text{cut})$ is the value of the rules in the cut between m and S. For any subset $S \subset X_i$ that does not include m, we need to consider the values of rules that include $\neg m$ as a literal in the pattern. Also, when computing the reserve, the payoff x_m will not contribute to group S. Therefore, together with the running intersection property as argued above, we can show that $r_i(S) = r_j(S) - v(\text{cut})$.

Join nodes (i with left child j and right child k) For any given set $S \subseteq X_i$, consider the r-values of that set at j and k. If $\arg r_j(S)$ or $\arg r_k(S)$ includes agents not in S, then $argr_j(S)$ and $argr_k(S)$ will be disjoint from each other due to the running intersection property. Therefore, we can decompose $\arg r_i(S)$ into three sets, $(\arg r_j(S) \setminus S)$ on the left, S in the middle, and $(\arg r_k(S) \setminus S)$ on the right. The reserve $r_j(S)$ will cover the excesses on the left and in the middle, whereas the reserve $r_k(S)$ will cover those on the right and in the middle, and so the excesses in the middle is double-counted. We adjust for the double-counting by subtracting the excesses in the middle from the sum of the two reserves $r_j(S)$ and $r_k(S)$.

□

Finally, note that each step in the computation of the r-values of each node i takes time at most exponential in the size of X_i, hence the algorithm runs in time exponential only in the treewidth of the graph.

5.4 Algorithm for Core Non-emptiness

We can extend the algorithm for CORE-MEMBERSHIP into an algorithm for CORE-NON-EMPTINESS. As described in section 2, whether the core is empty can be checked using the optimization program based on the balancedness condition (3). Unfortunately, that program has an exponential number of variables. On the other hand, the dual of the program has only n variables, and can be written as follows:

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & \sum_{i=1}^n x_i \\ \text{subject to} & x(S) \geq v(S), \quad \forall S \subseteq N \end{array} \tag{7}$$

By strong duality, optimal value of (7) is equal to optimal value of (4), the primal program described in section 2. Therefore, by the Bondereva-Shapley theorem, if the optimal value of (7) is greater than $v(N)$, the core is empty.

We can solve the dual program using the ellipsoid method with CORE-MEMBERSHIP as a separation oracle, i.e., a procedure for returning a constraint that is violated. Note that a simple extension to the CORE-MEMBERSHIP algorithm will allow us to keep track of the set T for which $e(T) < 0$ during the r-values computation, and hence we can return the inequality about T as the constraint violated. Therefore, CORE-NON-EMPTINESS can run in time polynomial in the running time of CORE-MEMBERSHIP, which in turn runs in time exponential only in the treewidth of the graph. Note that when the core is not empty, this program will return an outcome in the core.

6. CONCLUDING REMARKS

We have developed a fully expressive representation scheme for coalitional games of which the size depends on the complexity of the interactions among the agents. Our focus on general representation is in contrast to the approach taken in [3, 4]. We have also developed an efficient algorithm for the computation of the Shapley values for this representation. While CORE-MEMBERSHIP for MC-nets is coNP-complete, we have developed an algorithm for CORE-MEMBERSHIP that runs in time exponential only in the treewidth of the agent graph. We have also extended the algorithm to solve CORE-NON-EMPTINESS. Other than the algorithm for CORE-NON-EMPTINESS in [4] under the restriction of non-negative edge weights, and that in [2] for superadditive games when the value of the grand coalition is given, we are not aware of any explicit description of algorithms for core-related problems in the literature.

The work in this paper is related to a number of areas in computer science, especially in artificial intelligence. For example, the graphical interpretation of MC-nets is closely related to Markov random fields (MRFs) of the Bayes nets community. They both address the issue of of conciseness of representation by using the combinatorial structure of weighted hypergraphs. In fact, Kearns *et al.* first apply these idea to games theory by introducing a representation scheme derived from Bayes net to represent non-cooperative games [6]. The representational issues faced in coalitional games are closely related to the problem of expressing valuations in combinatorial auctions [5, 10]. The OR-bid language, for example, is strongly related to superadditivity. The question of the representation power of different patterns is also related to Boolean expression complexity [12]. We believe that with a better understanding of the relationships among these related areas, we may be able to develop more efficient representations and algorithms for coalitional games.

Finally, we would like to end with some ideas for extending the work in this paper. One direction to increase the conciseness of MC-nets is to allow the definition of equivalent classes of agents, similar to the idea of extending Bayes nets to probabilistic relational models. The concept of symmetry is prevalent in games, and the use of classes of agents will allow us to capture symmetry naturally and concisely. This will also address the problem of unpleasing assymetric representations of symmetric games in our representation.

Along the line of exploiting symmetry, as the agents within the same class are symmetric with respect to each other, we can extend the idea above by allowing functional description of marginal contributions. More concretely, we can specify the value of a rule as dependent on the number of agents of each relevant class. The use of functions will allow concise description of marginal diminishing returns (MDRs). Without the use of functions, the space needed to describe MDRs among n agents in MC-nets is $O(n)$. With the use of functions, the space required can be reduced to $O(1)$.

Another idea to extend MC-nets is to augment the semantics to allow constructs that specify certain rules cannot be applied simultaneously. This is useful in situations where a certain agent represents a type of exhaustible resource, and therefore rules that depend on the presence of the agent should not apply simultaneously. For example, if agent i in the system stands for coal, we can either use it as fuel for a power plant or as input to a steel mill for making steel, but not for both at the same time. Currently, to represent such situations, we have to specify rules to cancel out the effects of applications of different rules. The augmented semantics can simplify the representation by specifying when rules cannot be applied together.

7. ACKNOWLEDGMENT

The authors would like to thank Chris Luhrs, Bob McGrew, Eugene Nudelman, and Qixiang Sun for fruitful discussions, and the anonymous reviewers for their helpful comments on the paper.

8. REFERENCES

[1] H. L. Bodlaender. Treewidth: Algorithmic techniques and results. In *Proc. 22nd Symp. on Mathematical Foundation of Copmuter Science*, pages 19–36. Springer-Verlag LNCS 1295, 1997.

[2] V. Conitzer and T. Sandholm. Complexity of determining nonemptiness of the core. In *Proc. 18th Int. Joint Conf. on Artificial Intelligence*, pages 613–618, 2003.

[3] V. Conitzer and T. Sandholm. Computing Shapley values, manipulating value division schemes, and checking core membership in multi-issue domains. In *Proc. 19th Nat. Conf. on Artificial Intelligence*, pages 219–225, 2004.

[4] X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Math. Oper. Res.*, 19:257–266, May 1994.

[5] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proc. 16th Int. Joint Conf. on Artificial Intelligence*, pages 548–553, 1999.

[6] M. Kearns, M. L. Littman, and S. Singh. Graphical models for game theory. In *Proc. 17th Conf. on Uncertainty in Artificial Intelligence*, pages 253–260, 2001.

[7] J. Kleinberg, C. H. Papadimitriou, and P. Raghavan. On the value of private information. In *Proc. 8th Conf. on Theoretical Aspects of Rationality and Knowledge*, pages 249–257, 2001.

[8] C. Li and K. Sycara. Algoirthms for combinatorial coalition formation and payoff division in an electronic marketplace. Technical report, Robotics Insititute, Carnegie Mellon University, November 2001.

[9] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory.* Oxford University Press, New York, 1995.

[10] N. Nisan. Bidding and allocation in combinatorial auctions. In *Proc. 2nd ACM Conf. on Electronic Commerce*, pages 1–12, 2000.

[11] M. J. Osborne and A. Rubinstein. *A Course in Game Theory.* The MIT Press, Cambridge, Massachusetts, 1994.

[12] I. Wegener. *The Complexity of Boolean Functions.* John Wiley & Sons, New York, October 1987.

APPENDIX

We will formally show the correctness of the r-value computation in Algorithm 1 of introduce nodes and join nodes.

LEMMA 1. *The procedure for computing the r-values of introduce nodes in Algorithm 1 is correct.*

PROOF. Let node m be the newly introduced agent at i. Let U denote the set of agents in the subtree rooted at i. By the running intersection property, all interactions (the hyperedges) between m and U must be in node i. For all $S \subseteq X_i : m \in S$, let R denote $(U \setminus X_i) \cup S)$, and Q denote $(R \setminus \{m\})$.

$$\begin{aligned}
r_i(S) &= r(S, R) \\
&= \min_{T:S\subseteq T\subseteq R} e(T) \\
&= \min_{T:S\subseteq T\subseteq R} x(T) - v(T) \\
&= \min_{T:S\subseteq T\subseteq R} x(T \setminus \{m\}) + x_m - v(T \setminus \{m\}) - v(\text{cut}) \\
&= \left(\min_{T':S\setminus\{m\}\subseteq T'\subseteq Q} e(T')\right) + x_m - v(\text{cut}) \\
&= r_j(S) + x_m - v(\text{cut})
\end{aligned}$$

The argument for sets $S \subseteq X_i : m \notin S$ is symmetric except x_m will not contribute to the reserve due to the absence of m. □

LEMMA 2. *The procedure for computing the r-values of join nodes in Algorithm 1 is correct.*

PROOF. Consider any set $S \subseteq X_i$. Let U_j denote the subtree rooted at the left child, R_j denote $((U_j \setminus X_j) \cup S)$, and Q_j denote $(U_j \setminus X_j)$. Let U_k, R_k, and Q_k be defined analogously for the right child. Let R denote $(U \setminus X_i) \cup S)$.

$$\begin{aligned}
r_i(S) &= r(S, R) \\
&= \min_{T:S\subseteq T\subseteq R} x(T) - v(T) \\
&= \min_{T:S\subseteq T\subseteq R} \Big(x(S) + x(T \cap Q_j) + x(T \cap Q_k) \\
&\quad - v(S) - v(\text{cut}(S, T \cap Q_j) - v(\text{cut}(S, T \cap Q_k)\Big) \\
&= \min_{T:S\subseteq T\subseteq R} \Big(x(T \cap Q_j) - v(\text{cut}(S, T \cap Q_j))\Big) \\
&\quad + \min_{T:S\subseteq T\subseteq R} \Big(x(T \cap Q_k) - v(\text{cut}(S, T \cap Q_k))\Big) \\
&\quad + (x(S) - v(S)) \qquad (*) \\
&= \min_{T:S\subseteq T\subseteq R} \Big(x(T \cap Q_j) + x(S) - v(\text{cut}(S, T \cap Q_j)) - v(S)\Big) \\
&\quad + \min_{T:S\subseteq T\subseteq R} \Big(x(T \cap Q_k) + x(S) - v(\text{cut}(S, T \cap Q_k)) - v(S)\Big) \\
&\quad - (x(S) - v(S)) \\
&= \min_{T:S\subseteq T\subseteq R} e(T \cap R_j) + \min_{T:S\subseteq T\subseteq R} e(T \cap R_k) - e(S) \\
&= \min_{T':S\subseteq T'\subseteq R_j} e(T') + \min_{T'':S\subseteq T\subseteq R_k} e(T'') - e(S) \\
&= r_j(S) + r_k(S) - e(S)
\end{aligned}$$

where (*) is true as $T \cap Q_j$ and $T \cap Q_k$ are disjoint due to the running intersection property of tree decomposition, and hence the minimum of the sum can be decomposed into the sum of the minima. □

Boosting as a Metaphor for Algorithm Design

Kevin Leyton-Brown, Eugene Nudelman,
Galen Andrew, Jim McFadden, and Yoav Shoham
{kevinlb;eugnud;galand;jmcf;shoham}@cs.stanford.edu

Stanford University, Stanford CA 94305

Abstract. Hard computational problems are often solvable by multiple algorithms that each perform well on different problem instances. We describe techniques for building an algorithm portfolio that can outperform its constituent algorithms, just as the aggregate classi ers learned by boosting outperform the classi ers of which they are composed. We also provide 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 combinatorial auction winner determination problem, showing that our portfolio outperforms the state-of-the-art algorithm by a factor of three.[1]

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; in 1976 Rice dubbed this the "algorithm selection problem" [13]. 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 re nement, 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.

[1] This work has previously been published as a two-page extended abstract [9].

* This work is generously supported by DARPA grant F30602-00-2-0598.

1.1 The Boosting Metaphor

Boosting is a machine learning paradigm due to Schapire [17] and widely studied since. Although this paper 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 classi ers can be combined to form an accurate ensemble when the classi ers' areas of effectiveness are suf ciently uncorrelated.
2. New classi ers should be trained on problems on which the current aggregate classi er performs poorly.

In this paper, 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 suf ciently 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 5.

1.2 Case Study: Combinatorial Auctions (Weighted Set Packing)

To discuss the effectiveness of an algorithm design methodology, it is necessary to perform a case study. We chose to consider the combinatorial auction winner determination problem (WDP), and made use of runtime prediction techniques and runtime data from our previous work [10]. However, it must be emphasized that none of the techniques we propose here are particular to this problem domain. The full version of this paper will also consider other domains; in particular, we have had some positive initial results building portfolios for SAT.

Combinatorial auctions provide a general framework for allocation problems among self-interested agents by allowing bids for bundles of goods. WDP is a weighted set packing problem (SPP): the goal is to choose a non-con ictin g subset of bids maximizing the seller's revenue. SPP is $\mathcal{NP}$-Complete, and also inapproximable within a constant factor (cf. [15]). Let n be the number of goods, and m be the number of bids. A bid is a pair $< S_i, p_i >$, where $S_i \subseteq \{1, \ldots, n\}$ is the set of goods requested by bid i, and p_i is that bid's price offer. WDP can be formulated as the following integer program:

$$\begin{array}{rll} \text{maximize:} & \sum_{i=1}^{m} x_i p_i & \\ \text{subject to:} & \sum_{i \mid g \in S_i} x_i \leq 1 & \forall g \\ & x_i \in \{0, 1\} & \forall i \end{array}$$

We consider three algorithms for solving WDP: ILOG's CPLEX package;GL (Gonen-Lehmann) [5], a simple branch-and-bound algorithm with CPLEX's LP solver as its heuristic; and CASS [2], a more complex branch-and-bound algorithm with a non-LP heuristic. Unfortunately, we were unable to get access to CABOB [16], another widely-cited WDP algorithm.

1.3 Overview

In the next three sections we give general methods for our boosting analogy in algorithm design. In section 2 we present a methodology for constructing algorithm portfolios and show some results from our case study. We go on in section 3 to offer practical extensions to our methodology, including techniques for avoiding the computation of costly features, trading off between accuracy on hard and easy instances, and building models when runtime data is capped at some maximum running time. In section 4 we consider the empirical evaluation of portfolios, and describe a method for using a learned model of runtime to generate a test distribution that will be hard for a portfolio. Similar techniques can be used to generate instances that score highly on a given "realism" metric. Finally, section 5 discusses our design choices and compares them to the choices made in related work.

2 Algorithm Portfolios

Our previous work [10] demonstrated that statistical regression can be used to learn surprisingly accurate algorithm-speci c models of the empirical hardness of given distributions of problem instances. In short, the method proposed in that work is:

1. Use domain knowledge to select features of problem instances that might be indicative of runtime.
2. Generate a set of problem instances from the given distribution, and collect runtime data for the algorithm on each instance.
3. Use regression to learn a real-valued function of the features that predicts runtime.

Given this existing technique for predicting runtime, we now 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

This technique is powerful, but deceptively simple. For discussion and comparison with other approaches in the literature, please see section 5.1. As we will demonstrate in our case study, such portfolios can dramatically outperform the algorithms of which they are composed.

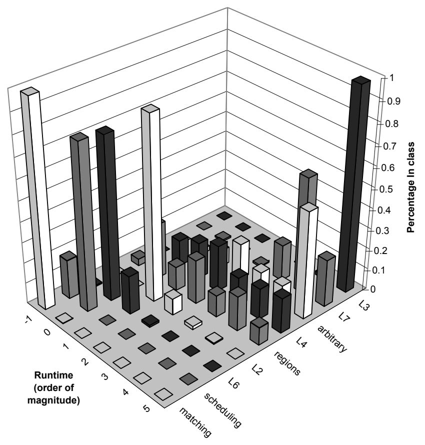


Fig. 1. Gross Hardness for CPLEX (from [10])

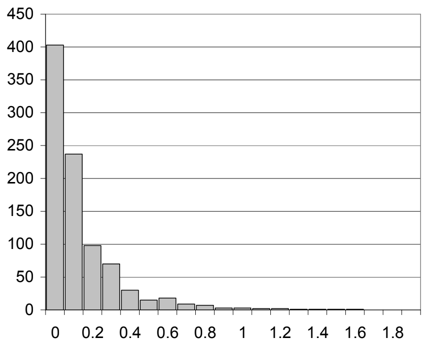


Fig. 2. Mean Absolute Error (from [10])

2.1 Case Study: Experimental Setup

We performed our case study using data collected in our past work [10], which we recap brie y in this section. All of our results focused on problems of a x ed size: numbers of goods and non-dominated bids were held constant to 256 and 1000 respectively.[2] Our instance distribution involved making a uniform choice between nine of the distributions from the Combinatorial Auction Test Suite (CATS) [11], and randomly choosing parameters for each instance. The complete dataset was composed of about 4500 instances. For each instance we collected runtime data for CPLEX 7.1, and computed 25 features that fall roughly into four categories:

1. Norms of the linear programming slack vector (integrality of the LP relaxation of the IP)
2. Deviations of prices
3. Node statistics of the Bid-Good bipartite graph
4. Various statistics of the Bid graph (effectively, the problem's constraint graph)

All data was collected on 550 MHz Pentium Xeon machines running Linux 2.2; over 3 years of CPU time was spent gathering this data. Fig. 1 shows a 3D histogram of the distribution of hard instances across our dataset. Observe that CPLEX's runtime varied by seven orders of magnitude even though the number of goods and bids was held constant. Also, there is considerable variation within most of the distributions.

Using quadratic regression, we were able to build very accurate models of the logarithm of runtime. Fig. 2 shows a histogram of the mean absolute error in predicting the log of CPLEX's runtime observed on test set instances. Since our methodology relies

[2] In a separate research effort, we are in the process of extending the work from [10] to models of variable problem size; when these models become available it will be possible to extend the techniques presented in this paper without any modi cation.

on machine learning, we split the data into training, validation, and test sets. We report our portfolio runtimes only on the test set that was never used to train or evaluate models. An error of 1 in predicting the log means that runtime was mispredicted by a factor of 10, or roughly that an instance was misclassi ed by one of the bins in Fig. 1; observe that nearly all prediction errors are less than 1.

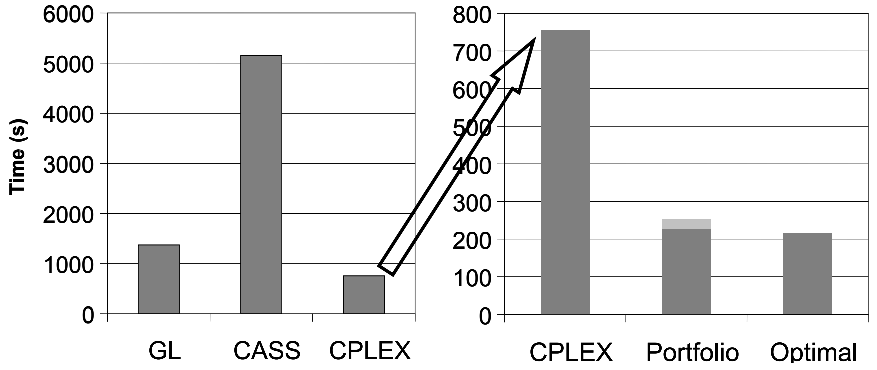


Fig. 3. Algorithm and Portfolio Runtimes

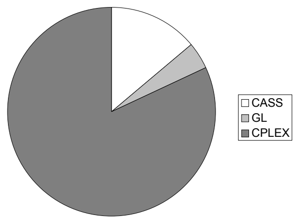


Fig. 4. Optimal

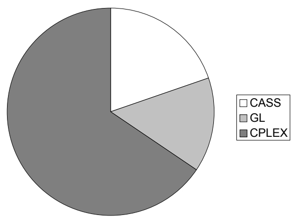


Fig. 5. Selected

2.2 Case Study Results

We now turn to new results. First, we used the methodolody described in section 2.1 to build regression models for two new algorithms (GL and CASS). Fig. 3 compares the average runtimes of our three algorithms (CPLEX, CASS, GL) to that of the portfolio[3]. Note that 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

[3] Note the change of scale on the graph, and the repeated CPLEX bar

cost of computing features, our portfolio's selections take only 5% longer to run than the optimal selections.

Figs. 4 and 5 show the frequency with which each algorithm is selected in the ideal portfolio and in our portfolio. They illustrate the quality of our algorithm selection and the relative value of the three algorithms. 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.

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 work even better in other settings.

3 Extending our Portfolio Methodology

Once it has been demonstrated that algorithm portfolios can offer signi cant speedups over winner-take-all algorithm selection, it is worthwhile to consider modi cations to the methodology that make it more useful in practice. Speci cally, we describe methods for reducing the amount of time spent computing features, transforming the response variable, and capping runs of some or all algorithms.

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 suf cient 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 signi cantly longer to compute than each feature in S_j.[4] The portfolio can start by computing the easiest features, and iteratively compute the next set only if the expected bene t 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 rst 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 $\text{argmin}_i\, M_k^i$.

[4] We assume here that features will have low runtime variance. We have found this assumption to hold in our case study. If feature runtime variance makes it dif cult to partition the features into time complexity sets, smart feature computation is more dif cult.

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 t.

Given an instance x, the portfolio now works as follows:

4. For $j = 1$ to l
   (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.

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, they appropriately penalize 20 seconds of error equally on instances that take 1 second or 10 hours to run. However, another reasonable goal may be to perform well on every instance regardless of its hardness; in this case, relative error is more appropriate. Let r_i^p and r_i^* be the portfolio's runtime and the optimal runtime respectively on instance i, and n be the number of instances. One measure that gives an insight into the portfolio's relative error is *percent optimal*:

$$\sum_{i|r_i^p = r_i^*} \frac{1}{n}.$$

Another measure of relative error is *average percent suboptimal*:

$$\frac{1}{n} \sum_i \frac{r_i^p - r_i^*}{r_i^*}.$$

Taking a logarithm of runtime is a simple way to equalize the importance of relative error on easy and hard instances. Thus, models that predict a log of runtime help to improve the average percent suboptimal, albeit at some expense in terms of the portfolio's average runtime. In Figure 6 (overleaf) we show three different functions; linear (identity) and log are the extreme values; clearly, many functions can fall in between. The functions are normalized by their maximum value, since this does not affect regression, but allows us to better visualize their effect. In our case study (section 3.4) we found that the cube root function was particularly effective.

3.3 Capping Runs

The methodology of section 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 signi cantly greater than the minimum of the algorithms' runtimes; if not, it might still be preferable to sacri ce 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.

3.4 Case Study Results

Fig. 7 shows the performance of the smart feature computation discussed in section 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.

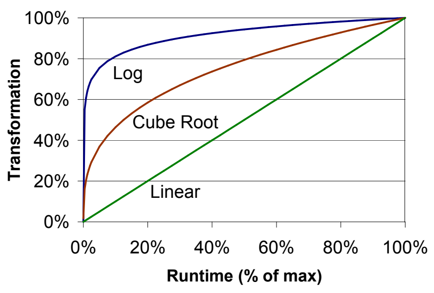


Fig. 6. Transformation F'ns (Normalized)

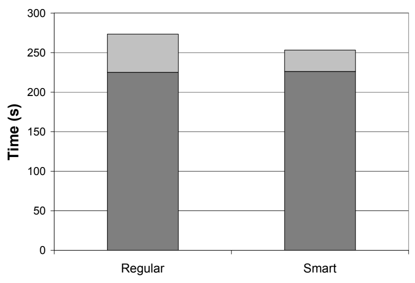


Fig. 7. Smart Features

	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 1. Portfolio Results

Table 1 shows the effect of our response variable transformations on average runtime, percent optimal and average percent suboptimal. The rst row has results that would be obtained by a perfect portfolio. As discussed in section 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.

4 Focused Algorithm Design

Once we have decided to select among existing algorithms using a portfolio approach, it is necessary to reexamine the way we design and evaluate algorithms. Since the purpose of designing new algorithms is to reduce the time that it will take to solve problems, designers should aim to produce new algorithms that complement an existing portfolio. First, it is essential to choose a distribution D that re ects 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 classi ers should be trained on instances that are hard for the existing ensemble, in the proportion that they occur in the original training set.

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. 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 [1]: 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:[5]

1. Create a hardness model H_p with features $\overrightarrow{p}$, and normalize it to create a pdf, h_p.
2. Generate a large number of instances from $D \cdot h_p$.
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

[5] 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.

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 dif cult 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 $h_{f,i}$.[6] 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.
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.

4.2 Inducing Realistic Distributions

It is important for our portfolio methodology that we begin with a "realistic" D: that is, a distribution accurately re ecting 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 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. This can allow us to make informed choices when setting the parameters of instance generators, and also to discard less realistic data after it has been generated. Note that when inducing hard distributions a hardness model had to be used because it was infeasible to score each sample by actual portfolio runtime. In the case of inducing realistic distributions this is no longer a problem, because the realism function *can* be evaluated on each sample. Therefore, our rejection sampling technique is guaranteed to generate instances with increased average realism scores. The use of parameter-space models r_p can still improve performance by reducing the number of samples needed for obtaining good results.

[6] However, the case study results presented in gs. 8–10 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.

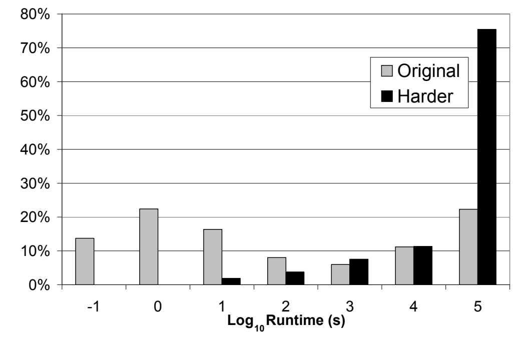
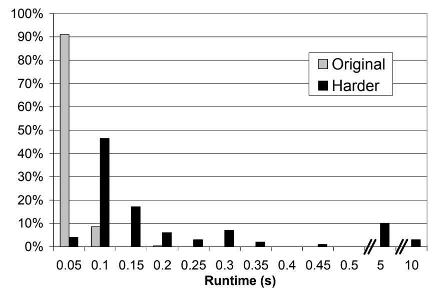


Fig. 8. Inducing Harder Distributions



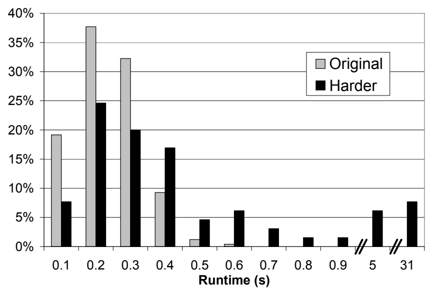


Fig. 9. Matching

Fig. 10. Scheduling

4.3 Case Study 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 [10], it is quite easy for our technique to generate harder instances. These results are presented in g. 8. 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 it happens, there has been ongoing discussion in the WDP literature about whether those CATS distributions [11] that are relatively easy could be con gured to be harder (see e.g., [4, 16]). We consider two easy distributions with low variance from CATS, *matching* and *scheduling*, and show that they indeed can be made harder than originally proposed. Figures 9 and 10 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.

5 Discussion and Related Work

Although it is helpful, our analogy to boosting is clearly not perfect. One key difference lies in the way components are aggregated: classi ers 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.

5.1 Algorithm Selection

It has long been understood that algorithm performance can vary substantially across different classes of problems. Rice [13] was the rst to formalize algorithm selection as a computational problem, framing it in terms of function approximation. Broadly, he identi ed the goal of selecting a mapping $S(x)$ from the space of instances to the space of algorithms, to maximize some performance measure 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 x ed, 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 the data from our case study 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 [3] 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 dif cult 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.

Classi cation Since algorithm selection is fundamentally discriminative—it entails choosing the algorithm that will exhibit minimal runtime—classi cation is an obvious approach to consider. Any standard classi cation 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 classi cation algorithms use the wrong error metric: they penalize misclassi cations 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 misclassi cations 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 t to minimize root mean squared error.

A second classi cation 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. [6, 14] used classi cation 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 classi cation algorithm to predict runtime. First, the learning algorithm does not use an error function that penalizes large misclassi cations (off by more than one bin) more heavily than small misclassi cations (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 arti cial. Instances with runtimes lying very close to a boundary are likely to be misclassi ed even by a very accurate model, making accurate models harder to learn.

Markov Decision Processes Perhaps most related to our paper is work by Lagoudakis and Littman ([7, 8]). 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; in our case study 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 can also be described in an MDP framework, with each action (choice of algorithm) leading to a terminal state, and reward equal to the negative of runtime. Optimal policy selection is trivial given a good value function; thus the key to success is good value estimation. Our approach emphasizes making the value functions—that is, models of runtime—explicit, since this provides the best defense against good but fragile policies. We do not describe our models as MDPs because the framework is redundant in the absence of sequential decision-making.

Different Regression Approaches Lobjois and Lemaˆtre [12] 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.

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. ([18]), which describes a method of generating SAT instances near the phase transition threshold, which are known to be hard for most SAT solvers.

6 Conclusions

Just as boosting allows weak classi ers to work together effectively, algorithms can be combined into portfolios to build a whole greater than the sum of its parts. First, we have described how to build such portfolios. Our techniques can be elaborated to reduce the cost of computing features, to reduce the time spent gathering training data by capping runs, and to strike the right balance between the penalties for mispredicting easy and hard instances. Second, we argued that algorithm design should focus on problem instances upon which a portfolio of existing algorithms spends most of its time. We have provided techniques for inducing such distributions, and also for re ning distributions to emphasize instances that have high scores on a given 'realism' function. We performed a case study on combinatorial auctions, and showed that a portfolio composed of CPLEX and two older—and generally *much* slower—algorithms outperformed CPLEX alone by about a factor of 3. In future work, we aim to perform case studies of our methodology on other hard problems; our rst effort in this direction is a portfolio of 10 algorithms which we have entered in the 2003 SAT competition.

Acknowledgments

Thanks to Ryan Porter, Carla Gomes and Bart Selman for helpful discussions. This work was supported by DARPA grant F30602-00-2-0598, the Intelligent Information Systems Institute at Cornell, and a Stanford Graduate Fellowship.

References

1. A. Doucet, N. de Freitas, and N. Gordon(ed.). *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.
2. Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *IJCAI*, 1999.
3. C. Gomes and B. Selman. Algorithm portfolios. *Arti cial Intelligence*, 126(1-2):43–62, 2001.
4. R. Gonen and D. Lehmann. Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. In *ACM Conference on Electronic Commerce*, 2000.
5. R. Gonen and D. Lehmann. Linear programming helps solving large multi-unit combinatorial auctions. Technical Report TR-2001-8, Leibniz Center for Research in Computer Science, April 2001.
6. E. Horvitz, Y. Ruan, C. Gomes, H. Kautz, B. Selman, and M. Chickering. A Bayesian approach to tackling hard computational problems. In *UAI*, 2001.
7. M. Lagoudakis and M. Littman. Algorithm selection using reinforcement learning. In *ICML*, 2000.
8. M. Lagoudakis and M. Littman. Learning to select branching rules in the DPLL procedure for satis ability . In *LICS/SAT*, 2001.
9. K. Leyton-Brown, E. Nudelman, G. Andrew, J. McFadden, and Y. Shoham. A portfolio approach to algorithm selection. In *IJCAI*, 2003.
10. K. Leyton-Brown, E. Nudelman, and Y. Shoham. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In *CP*, 2002.
11. K. Leyton-Brown, M. Pearson, and Y. Shoham. Towards a universal test suite for combinatorial auction algorithms. In *ACM EC*, 2000.
12. L. Lobjois and M. Lemaˆtre. Branch and bound algorithm selection by performance prediction. In *AAAI*, 1998.
13. J. R. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.
14. Y. Ruan, E. Horvitz, and H. Kautz. Restart policies with dependence among runs: A dynamic programming approach. In *CP*, 2002.
15. T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *IJCAI*, 1999.
16. T. Sandholm, S. Suri, A. Gilpin, and D. Levine. CABOB: A fast optimal algorithm for combinatorial auctions. In *IJCAI*, 2001.
17. R. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.
18. B. Selman, D. G. Mitchell, and H. J. Levesque. Generating hard satis ability problems. *Arti cial Intelligence*, 81(1-2):17–29, 1996.

SATzilla: An Algorithm Portfolio for SAT*

Eugene Nudelman Alex Devkar Yoav Shoham
Department of Computer Science, Stanford University

Kevin Leyton-Brown Holger Hoos
Department of Computer Science, University of British Columbia

1 Introduction

Inspired by the success of recent work in the constraint programming community on typical-case complexity, in [3] we developed a new methodology for using machine learning to study empirical hardness of hard problems on realistic distributions. In [2] we demonstrated that this new approach can be used to construct practical algorithm portfolios. In brief, the fact that algorithms for solving $\mathcal{NP}$-hard problems are often relatively uncorrelated means that it is possible for a portfolio to outperform all of its constituent algorithms. However, such uncorrelation is a knife that cuts both ways: a portfolio that makes bad choices among its constituent algorithms will often have much *worse* performance than any of its constituent algorithms.

Our methodology can be outlined as follows:

Offline, as part of algorithm development:

1. Identify a target distribution of problem instances.
2. Select a set of algorithms having relatively uncorrelated runtimes on this distribution.
3. Using domain knowledge, identify features that characterize problem instances.
4. Compute features and determine algorithm running times.
5. Use regression to construct models of algorithms' runtimes.

Online, given an instance:

1. Compute feature values.
2. Predict each algorithm's running time using learned runtime models.
3. Run the algorithm predicted to be fastest.

*See [5] for a complete discussion of `SATzilla`

2 SATzilla

`SATzilla` is a portfolio of SAT solvers built according to the methodology described above. It includes the following solvers: `2clseq`, `Limmat`, `JeruSat`, `OKsolver`, `Relsat`, `Sato`, `Satz-rand`, `zChaff`, `eqSatz`, `Satzoo`, `kcnfs`, and `BerkMin`.

We began by assembling a broad library of about 5000 SAT instances, which we gathered from various public websites. We identified 83 features that could be computed quickly and that we felt might be useful for predicting runtime. We computed these features for our set of SAT instances, dropped some features that were highly correlated, and were left with 56 distinct features. In order to keep feature values to sensible ranges, as appropriate we normalized features by the total number of clauses or number of variables. We also computed runtimes for each algorithm on each of our SAT instances. Given our features and runtime data, we had a well-defined supervised learning problem. We built models using ridge regression, a machine learning technique that finds a linear model (a hyperplane in feature space) that minimizes a combination of root mean squared error and a penalty term for large coefficients. To yield better models, we ignored all instances that were solved by all algorithms, by no algorithms, or as a side-effect of feature computation.

Upon execution, `SATzilla` begins by running a UBCSAT [6] implementation of `WalkSat` for 30 seconds. In our experience, this step helps to filter out easy satisfiable instances. Next, `SATzilla` runs the `Hypre`[1] preprocessor, which uses hyper-resolution to reason about binary clauses. This step is often able to dramatically shorten the formula, often resulting in search problems that are easier for DPLL-style solvers. Perhaps more importantly, the simplification "cleans up" instances, allowing the subsequent analysis of

* This work is generously supported by DARPA grant F30602-00-2-0598.

their structure to better reflect the problem's combinatorial "core." Third, `SATzilla` computes its 56 features. Sometimes, a feature can actually solve the problem; if this occurs, execution stops. Some features can also take an inordinate amount of time, particularly with very large inputs. To prevent feature computation from consuming all of our allotted time, certain features run only until a timeout is reached, at which point `SATzilla` gives up on computing the given feature. Fourth, `SATzilla` evaluates a regression model of each algorithm in order to compute a prediction of that algorithm's running time. If some of the features have timed out, a different model is used, which does not involve the missing feature and which was trained only on instances where the same feature timed out. Finally, `SATzilla` runs the algorithm with the best predicted runtime until the instance is solved or the allotted time is used up.

3 Features

Space restrictions prevent us from going into great detail about all elements of `SATzilla`. We choose to use our remaining space to give an overview of the features used by `SATzilla`.

The features can be roughly categorized into 9 groups. The first one captures problem size, measured in the number of clauses, variables, and their ratio. The next three groups correspond to 3 different constraint graphs associated with each SAT instances. Variable-Clause Graph is a bipartite graph representing which variables participate in which clause. Variable Graph has nodes representing variables, and an edge between any variables that occur in a clause together. Conflict Graph (CG) has nodes representing clauses, and an edge between two clauses whenever they share a negated literal. For all graphs we compute various node degree statistics. For CG we also compute statistics of clustering coefficients, defined, for each node, as the number of edges among its neighbors divided by $k(k-1)/2$, where k is the number of neighbors. The fifth group measures the balance of an instance in several different respects: the number of unary, binary, and ternary clauses, statistics of the amount of positive versus negative occurrences of variables within clauses and per variable. The sixth group measures the proximity of the instance to a Horn formula by computing the fraction of clauses that are Horn, and statistics over variables occurring in a Horn clause. These groups are motivated by known heuristics and tractable subclasses. The seventh group of features is obtained by solving a linear programming relaxation of an integer program representing the current SAT instance (a feature that can sometimes solve the SAT instance). Often, for integer programs, proximity of the LP relaxation solution to an integral solution is anticorrelated with hardness. We compute statistics of the integer slacks, as well as the actual objective value and fraction of variables set to an integer. The eighth group tries to estimate the hardness of the search space for a DPLL-type solver. For that we run DPLL procedure to a small depth and measure the number of unit propagations done at various depths. We also estimate the size of the search space [4] by randomly setting variables and then doing unit propagation until a contradiction is found. Our final group of features uses two local search algorithms, GSAT and SAPS [6]. We run both algorithms many times, each time continuing the search trajectory until a plateau cannot be escaped within a given number of steps. We then average various statistics collected during each run. It is interesting to note that local-search probing features are important to the models for most of `SATzilla`'s algorithms, even though all of these solvers are DPLL-based.

References

[1] F. Bacchus and J. Winter. Effective preprocessing with hyper-resolution and equality reduction. In *SAT*, 2003.

[2] K. Leyton-Brown, E. Nudelman, G. Andrew, J. McFadden, and Y. Shoham. Boosting as a metaphor for algorithm design. In *CP*, 2003.

[3] K. Leyton-Brown, E. Nudelman, and Y. Shoham. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In *CP*, 2002.

[4] L. Lobjois and M. Lemaître. Branch and bound algorithm selection by performance prediction. In *AAAI*, 1998.

[5] E. Nudelman, A. Devkar, Y. Shoham, and K. Leyton-Brown. Understanding random SAT: Beyond the clauses-to-variables ratio. Under review.

[6] D. Tompkins and H. Hoos. UBCSAT: An implementation and experimentation environment for SLS algorithms for SAT and MAX-SAT. In *SAT*, 2004.