Computer Science and Game Theory: A Brief Survey

Joseph Y. Halpern*
Cornell University
Computer Science Department
Ithaca, NY 14853
halpern@cs.cornell.edu
http://www.cs.cornell.edu/home/halpern

May 18, 2007

1 Introduction

There has been a remarkable increase in work at the interface of computer science and game theory in the past decade. Game theory forms a significant component of some major computer science conferences (see, for example, [Kearns and Reiter 2005; Sandholm and Yakoo 2003]); leading computer scientists are often invited to speak at major game theory conferences, such as the World Congress on Game Theory 2000 and 2004. In this article I survey some of the main themes of work in the area, with a focus on the work in computer science. Given the length constraints, I make no attempt at being comprehensive, especially since other surveys are also available, including [Halpern 2003; Linial 1994; Papadimitriou 2001], and a comprehensive survey book will appear shortly [Nisan et al. 2007].

The survey is organized as follows. I look at the various roles of computational complexity in game theory in Section 2, including its use in modeling bounded rationality, its role in mechanism design, and the problem of computing Nash equilibria. In Section 3, I consider a game-theoretic problem that originated in the computer science literature, but should be of interest to the game theory community: computing the *price of anarchy*, that is, the cost of using decentralizing solution to a problem. In Section 4 I consider interactions between distributed computing and game theory. I conclude in Section 6 with a discussion of a few other topics of interest.

2 Complexity Considerations

The influence of computer science in game theory has perhaps been most strongly felt through complexity theory. I consider some of the strands of this research here. There are a numerous basic texts on complexity theory that the reader can consult for more background on notions like NP-completeness and finite automata, including [Hopcroft and Ullman 1979; Papadimitriou 1994a].

*Supported in part by Work supported in part by NSF under grants CTC-0208535 and ITR-0325453, by ONR under grant N00014-02-1-0455, by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the ONR under grants N00014-01-1-0795 and N00014-04-1-0725, and by AFOSR under grant F49620-02-1-0101. Thanks to Larry Blume, Christos Papadimitriou, Éva Tardos, and Moshe Tennenholtz for useful comments.

Bounded Rationality: One way of capturing bounded rationality is in terms of agents who have limited computational power. In economics, this line of research goes back to the work of Neyman [1985] and Rubinstein [1986], who focused on finitely repeated prisoner's dilemma. In n-round finitely repeated prisoner's dilemma, there are 2^{2^n-1} strategies (since a strategy is a function from histories to $\{$cooperate, defect$\}$, and there are clearly $2^n - 1$ histories of length $< n$). Finding a best response to a particular move can thus potentially be difficult. Clearly people do not find best responses by doing extensive computation. Rather, they typically rely on simple heuristics, such as "Tit for Tat" [Axelrod 1984]. Such heuristics can often be captured by finite automata; both Neyman and Rubinstein thus focus on finite automata playing repeated prisoners dilemma. Two computer scientists, Papadimitriou and Yannakakis [1994], showed that if both players in an n-round prisoners dilemma are finite automata with at least 2^{n-1} states, then the only equilibrium is the one where they defect in every round. This result says that a finite automaton with exponentially many states can compute best responses in prisoners dilemma.

We can then model bounded rationality by restricting the number of states of the automaton. Neyman [1985] showed, roughly speaking, that if the two players in n-round prisoner's dilemma are modeled by finite automata with a number of states in the interval $[n^{1/k}, n^k]$ for some k, then collaboration can be approximated in equilibrium; more precisely, if the payoff for (cooperate,cooperate) is (3,3) there is an equilibrium in the repeated game where the average payoff per round is greater than $3 - \frac{1}{k}$ for each player. Papadimitriou and Yannakakis [1994] sharpen this result by showing that if at least one of the players has fewer than $2^{c_\epsilon n}$ states, where $c_\epsilon = \frac{\epsilon}{12(1+\epsilon)}$, then for sufficiently large n, then there is an equilibrium where each player's average payoff per round is greater than $3 - \epsilon$. Thus, computational limitations can lead to cooperation in prisoner's dilemma.

There have been a number of other attempts to use complexity-theoretic ideas from computer science to model bounded rationality; see Rubinstein [1998] for some examples. However, it seems that there is much more work to be done here.

Computing Nash Equilibrium: Nash [1950] showed every finite game has a Nash equilibrium in mixed strategies. But how hard is it to actually find that equilibrium? On the positive side, there are well known algorithms for computing Nash equilibrium, going back to the classic Lemke-Howson [1964] algorithm, with a spate of recent improvements (see, for example, [Govindan and Wilson 2003; Blum et al. 2003; Porter et al. 2004]). Moreover, for certain classes of games (for example, symmetric games [Papadimitriou and Roughgarden 2005]), there are known to be polynomial-time algorithms. On the negative side, many questions about Nash equilibrium are known to be NP-hard. For example, Gilboa and Zemel [1989] showed that, for a game presented in normal form, deciding whether there exists a Nash equilibrium where each player gets a payoff of at least r is NP-complete. (Interestingly, Gilboa and Zemel also show that computing whether there exists a *correlated* equilibrium [Aumann 1987] where each player gets a payoff of at least r is computable in polynomial time. In general, questions regarding correlated equilibrium seem easier than the analogous questions for Nash equilibrium; see also [Papadimitriou 2005; Papadimitriou and Roughgarden 2005] for further examples.) Chu and Halpern [2001] prove similar NP-completeness results if the game is represented in extensive form, even if all players have the same payoffs (a situation which arises frequently in computer science applications, where we can view the players as agents of some designer, and take the payoffs to be the designer's payoffs). Conitzer and Sandholm [2003] give a compendium of hardness results for various questions one can ask about Nash equilibria.

Nevertheless, there is a sense in which it seems that the problem of finding a Nash equilibrium is easier than typical NP-complete problems, because every game is guaranteed to have a Nash equilibrium. By way of contrast, for a typical NP-complete problem like propositional satisfiability, whether or not a propositional formula is satisfiable is not known. Using this observation, it can be shown that if finding a Nash equilibrium is NP-complete, then NP = coNP. Recent work has in a sense has completely characterized the complexity of finding a Nash equilibrium in normal-form games: it is a *PPPAD-complete* problem [Chen and Deng 2006; Daskalis, Goldberg, and Papadimitriou 2006]. PPAD stands for "polynomial party argument (directed case)"; see [Papadimitriou 1994b] for a formal definition and examples of other PPAD problems. It is believed that PPAD-complete problems are not solvable in polynomial time, but are simpler than NP-complete problems, although this remains an open problem. See [Papadimitriou 2007] for an overview of this work.

Algorithmic Mechanism Design: The problem of mechanism design is to design a game such that the agents playing the game, motivated only by self-interest, achieve the designer's goals. This problem has much in common with the standard computer science problem of designing protocols that satisfy certain specifications (for example, designing a distributed protocol that achieves Byzantine agreement; see Section 4). Work on mechanism design has traditionally ignored computational concerns. But Kfir-Dahav, Monderer, and Tennenholtz [2000] show that, even in simple settings, optimizing social welfare is NP-hard, so that perhaps the most common approach to designing mechanisms, applying the Vickrey-Groves-Clarke (VCG) procedure [Clarke 1971; Groves 1973; Vickrey 1961], is not going to work in large systems. We might hope that, even if we cannot compute an optimal mechanism, we might be able to compute a reasonable approximation to it. However, as Nisan and Ronen [2000, 2001] show, in general, replacing a VCG mechanism by an approximation does not preserve truthfulness. That is, even though truthfully revealing one's type is an optimal strategy in a VCG mechanism, it may no longer be optimal in an approximation. Following Nisan and Ronen's work, there has been a spate of papers either describing computationally tractable mechanisms or showing that no computationally tractable mechanism exists for a number of problems, ranging from task allocation [Archer and Tardos 2001; Nisan and Ronen 2001] to costsharing for multicast trees [Feigenbaum et al. 2000] (where the problem is to share the cost of sending, for example, a movie over a network among the agents who actually want the movie) to finding low-cost paths between nodes in a network [Archer and Tardos 2002].

The problem that has attracted perhaps the most attention is *combinatorial auctions*, where bidders can bid on bundles of items. This becomes of particular interest in situations where the value to a bidder of a bundle of goods cannot be determined by simply summing the value of each good in isolation. To take a simple example, the value of a pair of shoes is much higher than that of the individual shoes; perhaps more interestingly, an owner of radio stations may value having a license in two adjacent cities more than the sum of the individual licenses. Combinatorial auctions are of great interest in a variety of settings including spectrum auctions, airport time slots (i.e., takeoff and landing slots), and industrial procurement. There are many complexity-theoretic issues related to combinatorial auctions. For a detailed discussion and references, see [Cramton et al. 2006]; I briefly discuss a few of the issues involved here.

Suppose that there are n items being auctioned. Simply for a bidder to communicate her bids to the auctioneer can take, in general, exponential time, since there are 2^n bundles. In many cases, we can identify a bid on a bundle with the bidder's valuation of the bundle. Thus, we can try to carefully design a bidding language in which a bidder can communicate her valuations succinctly. Simple information-theoretic arguments can be used to show that, for every bidding language, there will be valuations that will require length at least 2^n to express in that language. Thus, the best we can hope for is to design a language that can represent the "interesting" bids succinctly. See [Nisan 2006] for an overview of various bidding languages and their expressive power.

Given bids from each of the bidders in a combinatorial auction, the auctioneer would like to then determine the winners. More precisely, the auctioneer would like to allocate the m items in an auction so as to maximize his revenue. This problem, called the *winner determination problem*, is NP-complete in general, even in relatively simple classes of combinatorial auctions with only two bidders making rather restricted bids. Moreover, it is not even polynomial-time approximable, in the sense that there is no constant d and polynomial-time algorithm such that the algorithm produces an allocation that gives revenue that is at least $1/d$ of optimal. On the other hand, there are algorithms that provably find a good solution, seem to work well in practice, and, if they seem to taking too long, can be terminated early, usually with a good feasible solution in hand. See [Lehmann et al. 2006] for an overview of the results in this area.

In most mechanism design problems, computational complexity is seen as the enemy. There is one class of problems in which it may be a friend: voting. One problem with voting mechanisms is that of *manipulation* by voters. That is, voters may be tempted to vote strategically rather than ranking the candidates according to their true preferences, in the hope that the final outcome will be more favorable. This situation arises frequently in practice; in the 2000 election, American voters who preferred Nader to Gore to Bush were encouraged to vote for Gore, rather than "wasting" a vote on Nader. The classic Gibbard-Satterthwaite theorem [Gibbard 1973; Satterthwaite 1975] shows that, if there are at least three alternatives, then in any nondictatorial voting scheme (i.e., one where it is *not* the case that one particular voter dictates the final outcome, irrespective of how the others vote), there are preferences under which an agent is better off voting strategically. The hope is that, by constructing the voting mechanism appropriately, it may be computationally intractable to find a manipulation that will be beneficial. While finding manipulations for majority voting (the candidate with the most votes wins) is easy, there are well-known voting protocols for which manipulation is hard in the presence of three or more candidates. See [Conitzer et al. 2007] for a summary of results and further pointers to the literature.

Communication Complexity: *Communication complexity* [Kushilevitz and Nisan 1997] studies how much communication is needed for a set of n agents to compute the value of a function $f : \times_{i=1}^{n} \Theta_i \rightarrow X$, where each agent i knows $\theta_i \in \Theta_i$. To see the relevance of this to economics, consider, for example, the problem of mechanism design. Most mechanisms in the economics literature are designed so that agents truthfully reveal their preferences (think of θ_i as characterizing agent i's preferences here). However, in some settings, revealing one's full preferences can require a prohibitive amount of communication. For example, in a combinatorial auction of m items, revealing one's full preferences may require revealing what one would be willing to pay for each of the $2^m - 1$ possible bundles of items. Even if $m = 30$, this requires revealing more than one billion numbers. This leads to an obvious question: how much communication is required by various mechanisms? Nisan and Segal [2005] show that a standard approach for conducting combinatorial auctions, where prices are listed, agents are expected to make demands based on these prices, and then prices are adjusted (according to some pre-specified rule) based on demand, requires an exponential amount of communication for a certain class of valuations. This is among the first of preliminary steps to understanding the communication complexity of mechanisms; the general problem remains wide open.

3 The Price of Anarchy

In a computer system, there are situations where we may have a choice between invoking a centralized solution to a problem or a decentralized solution. By "centralized" here, I mean that each agent in the system is told exactly what to do and must do so; in the decentralized solution, each agent tries to optimize his own selfish interests. Of course, centralization comes at a cost. For one thing, there is a problem of enforcement. For another, centralized solutions tend to be more vulnerable to failure. On the other hand, a centralized solution may be more socially beneficial. How much more beneficial can it be?

Koutsoupias and Papadimitriou [1999] formalized this question by comparing the ratio of the social welfare of the centralized solution to that of the social welfare of the Nash equilibrium with the worst social welfare (assuming that the social welfare function is always positive). They called this ratio the *the price of anarchy*, and proved a number of results regarding the price of anarchy for a scheduling problem on parallel machines. Since the original paper, the price of anarchy has been studied in many settings, including traffic routing [Roughgarden and Tardos 2002], facility location games (e.g., where is the best place to put a factory) [Vetta 2002], and spectrum sharing (how should channels in a WiFi network be assigned) [Halldórsson et al. 2004].

To give a sense of the results, consider the traffic-routing context of Roughgarden and Tardos [2002]. Suppose that the travel time on a road increases in a known way with the congestion on the road. The goal is to minimize the average travel time for all drivers. Given a road network and a given traffic load, a centralized solution would tell each driver which road to take. For example, there could be a rule that cars with odd-numbered license plates take road 1, while those with even-numbered plates take road 2, to minimize congestion on either road. Roughgarden and Tardos show that the price of anarchy is unbounded if the travel time can be a nonlinear function of the congestion. On the other hand, if it is linear, they show that the price of anarchy is at most $4/3$.

The price of anarchy is but one way of computing the "cost" of using a Nash equilibrium. Others have been considered in the computer science literature. For example, Tennenholtz [2002] compares the *safety level* of a game—the optimal amount that an agent can guarantee himself, independent of what the other agents do—to what the agent gets in a Nash equilibrium, and shows, for interesting classes of games, including load-balancing games and first-price auctions, the ratio between the safety level and the Nash equilibrium is bounded. For example, in the case of first-price auctions, it is bounded by the constant e.

4 Game Theory and Distributed Computing

Distributed computing and game theory are interested in much the same problems: dealing with systems where there are many agents, facing uncertainty, and having possibly different goals. In practice, however, there has been a significant difference in emphasis in the two areas. In distributed computing, the focus has been on problems such as fault tolerance, asynchrony, scalability, and proving correctness of algorithms; in game theory, the focus has been on strategic concerns. I discuss here some issues of common interest.[1]

To understand the relevance of fault tolerance and asynchrony, consider the *Byzantine agreement* problem, a paradigmatic problem in the distributed systems literature. In this problem, there are assumed

[1] Much of the discussion in this section is taken from [Halpern 2003].

to be n soldiers, up to t of which may be faulty (the t stands for *traitor*); n and t are assumed to be common knowledge. Each soldier starts with an initial preference, to either attack or retreat. (More precisely, there are two types of nonfaulty agents—those that prefer to attack, and those that prefer to retreat.) We want a protocol that guarantees that (1) all *nonfaulty* soldiers reach the same decision, and (2) if all the soldiers are nonfaulty and their initial preferences are identical, then the final decision agrees with their initial preferences. (The condition simply prevents the obvious trivial solutions, where the soldiers attack no matter what, or retreat no matter what.)

This was introduced by Pease, Shostak, and Lamport [1980], and has been studied in detail since then; Chor and Dwork [1989], Fischer [1983], and Linial [1994] provide overviews. Whether the Byzantine agreement problem is solvable depends in part on what types of failures are considered, on whether the system is *synchronous* or *asynchronous*, and on the ratio of n to t. Roughly speaking, a system is synchronous if there is a global clock and agents move in lockstep; a "step" in the system corresponds to a tick of the clock. In an asynchronous system, there is no global clock. The agents in the system can run at arbitrary rates relative to each other. One step for agent 1 can correspond to an arbitrary number of steps for agent 2 and vice versa. Synchrony is an implicit assumption in essentially all games. Although it is certainly possible to model games where player 2 has no idea how many moves player 1 has taken when player 2 is called upon to move, it is not typical to focus on the effects of synchrony (and its lack) in games. On the other hand, in distributed systems, it is typically a major focus.

Suppose for now that we restrict to *crash failures*, where a faulty agent behaves according to the protocol, except that it might crash at some point, after which it sends no messages. In the round in which an agent fails, the agent may send only a subset of the messages that it is supposed to send according to its protocol. Further suppose that the system is synchronous. In this case, the following rather simple protocol achieves Byzantine agreement:

- In the first round, each agent tells every other agent its initial preference.
- For rounds 2 to $t + 1$, each agent tells every other agent everything it has heard in the previous round. (Thus, for example, in round 3, agent 1 may tell agent 2 that it heard from agent 3 that its initial preference was to attack, and that it (agent 3) heard from agent 2 that its initial preference was to attack, and it heard from agent 4 that its initial preferences was to retreat, and so on. This means that messages get exponentially long, but it is not difficult to represent this information in a compact way so that the total communication is polynomial in n, the number of agents.)
- At the end of round $t + 1$, if an agent has heard from any other agent (including itself) that its initial preference was to attack, it decides to attack; otherwise, it decides to retreat.

Why is this correct? Clearly, if all agents are correct and want to retreat (resp., attack), then the final decision will be to retreat (resp., attack), since that is the only preference that agents hear about (recall that for now we are considering only crash failures). It remains to show that if some agents prefer to attack and others to retreat, then all the nonfaulty agents reach the same final decision. So suppose that i and j are nonfaulty and i decides to attack. That means that i heard that some agent's initial preference was to attack. If it heard this first at some round $t' < t + 1$, then i will forward this message to j, who will receive it and thus also attack. On the other hand, suppose that i heard it first at round $t + 1$ in a message from i_{t+1}. Thus, this message must be of the form "i_t said at round t that ... that i_2 said at round 2 that i_1 said at round 1 that its initial preference was to attack." Moreover, the agents $i_1, \ldots, i_{t+1}$ must all be distinct. Indeed, it is easy to see that i_k must crash in round k before sending its message to i (but after sending its message to i_{k+1}), for $k = 1, \ldots, t$, for otherwise i must have gotten the message from i_k, contradicting the assumption that i first heard at round $t+1$ that some agent's initial preference was to attack. Since at most t agents can crash, it follows that i_{t+1}, the agent that sent the message to i, is not faulty, and thus sends the message to j. Thus, j also decides to attack. A symmetric argument shows that if j decides to attack, then so does i.

It should be clear that the correctness of this protocol depends on both the assumptions made: crash failures and synchrony. Suppose instead that *Byzantine* failures are allowed, so that faulty agents can deviate in arbitrary ways from the protocol; they may "lie", send deceiving messages, and collude to fool the nonfaulty agents in the most malicious ways. In this case, the protocol will not work at all. In fact, it is known that agreement can be reached in the presence of Byzantine failures iff $t < n/3$, that is, iff fewer than a third of the agents can be faulty [Pease et al. 1980]. The effect of asynchrony is even more devastating: in an asynchronous system, it is impossible to reach agreement using a deterministic protocol even if $t = 1$ (so that there is at most one failure) and only crash failures are allowed [Fischer et al. 1985]. The problem in the asynchronous setting is that if none of the agents have heard from, say, agent 1, they have no way of knowing whether agent 1 is faulty or just slow. Interestingly, there are randomized algorithms (i.e., behavior strategies) that achieve agreement with arbitrarily high probability in an asynchronous setting [Ben-Or 1983; Rabin 1983].

Byzantine agreement can be viewed as a game where, at each step, an agent can either send a message or decide to attack or retreat. It is essentially a game between two teams, the nonfaulty agents and the faulty agents, whose composition is unknown (at least by the correct agents). To model it as a game in the more traditional sense, we could imagine that the nonfaulty agents are playing against a new player, the "adversary". One of adversary's moves is that of "corrupting" an agent: changing its type from "nonfaulty" to "faulty". Once an agent is corrupted, what the adversary can do depends on the failure type being considered. In the case of crash failures, the adversary can decide which of a corrupted agent's messages will be delivered in the round in which the agent is corrupted; however, it cannot modify the messages themselves. In the case of Byzantine failures, the adversary essentially gets to make the moves for agents that have been corrupted; in particular, it can send arbitrary messages.

Why has the distributed systems literature not considered strategic behavior in this game? Crash failures are used to model hardware and software failures; Byzantine failures are used to model random behavior on the part of a system (for example, messages getting garbled in transit), software errors, and malicious adversaries (for example, hackers). With crash failures, it does not make sense to view the adversary's behavior as strategic, since the adversary is not really viewed as having strategic interests. While it would certainly make sense, at least in principle, to consider the probability of failure (i.e., the probability that the adversary corrupts an agent), this approach has by and large been avoided in the literature because it has proved difficult to characterize the probability distribution of failures over time. Computer components can perhaps be characterized as failing according to an exponential distribution (see [Babaoglu 1987] for an analysis of Byzantine agreement in such a setting), but crash failures can be caused by things other than component failures (faulty software, for example); these can be extremely difficult to characterize probabilistically. The problems are even worse when it comes to modeling random Byzantine behavior.

With malicious Byzantine behavior, it may well be reasonable to impute strategic behavior to agents (or to an adversary controlling them). However, it is often difficult to characterize the payoffs of a malicious agent. The goals of the agents may vary from that of simply trying to delay a decision to that of causing disagreement. It is not clear what the appropriate payoffs should be for attaining these goals. Thus, the distributed systems literature has chosen to focus instead on algorithms that are guaranteed to satisfy the specification without making assumptions about the adversary's payoffs (or nature's probabilities, in the case of crash failures).

Recently, there has been some working adding strategic concerns to standard problems in distributed computing (see, for example, [Halpern and Teague 2004]) as well as adding concerns of fault tolerance and asynchrony to standard problems in game theory (see, for example, [Eliaz 2002; Monderer and Tennenholtz 1999a; Monderer and Tennenholtz 1999b] and the definitions in the next section). This seems to be an area that is ripe for further developments. One such development is the subject of the next section.

5 Implementing Mediators

The question of whether a problem in a multiagent system that can be solved with a trusted mediator can be solved by just the agents in the system, without the mediator, has attracted a great deal of attention in both computer science (particularly in the cryptography community) and game theory. In cryptography, the focus on the problem has been on *secure multiparty computation*. Here it is assumed that each agent i has some private information x_i. Fix functions $f_1, \ldots, f_n$. The goal is have agent i learn $f_i(x_1, \ldots, x_n)$ without learning anything about x_j for $j \neq i$ beyond what is revealed by the value of $f_i(x_1, \ldots, x_n)$. With a trusted mediator, this is trivial: each agent i just gives the mediator its private value x_i; the mediator then sends each agent i the value $f_i(x_1, \ldots, x_n)$. Work on multiparty computation [Goldreich et al. 1987; Shamir et al. 1981; Yao 1982] provides conditions under which this can be done. In game theory, the focus has been on whether an equilibrium in a game with a mediator can be implemented using what is called *cheap talk*—that is, just by players communicating among themselves (cf. [Barany 1992; Ben-Porath 2003; Heller 2005; Urbano and Vila 2002; Urbano and Vila 2004]). As suggested in the previous section, the focus in the computer science literature has been in doing multiparty computation in the presence of possibly malicious adversaries, who do everything they can to subvert the computation, while in the game theory literature, the focus has been on strategic agents. In recent work, Abraham et al. [2006, 2007] considered deviations by both rational players, deviations by both rational players, who have preferences and try to maximize them, and players who can viewed as malicious, although it is perhaps better to think of them as rational players whose utilities are not known by the other players or mechanism designer. I briefly sketch their results here.[2]

The idea of tolerating deviations by coalitions of players goes back to Aumann [1959]; more recent refinements have been considered by Moreno and Wooders [1996]. Aumann's definition is essentially the following.

Definition 1 *$\vec{\sigma}$ is a k-*resilient$'$ equilibrium *if, for all sets C of players with $|C| \leq k$, it is not the case that there exists a strategy $\vec{\tau}$ such that $u_i(\vec{\tau}_C, \vec{\sigma}_{-C}) > u_i(\vec{\sigma})$ for all $i \in C$.* ∎

As usual, the strategy $(\vec{\tau}_C, \vec{\sigma}_{-C})$ is the one where each player $i \in C$ plays τ_i and each player $i \notin C$ plays σ_i. As the prime notation suggests, this is not quite the definition we want to work with. The trouble with this definition is that it suggests that coalition members cannot communicate with each other beyond agreeing on what strategy to use. Perhaps surprisingly, allowing communication can *prevent* certain equilibria (see [Abraham et al. 2007] for an example). Since we should expect coalition members to communicate, the following definition seems to capture a more reasonable notion of resilient equilibrium. Let the cheap-talk extension of a game Γ be, roughly speaking, the the game where players

[2]Much of the discussion in this section is taken from [Abraham et al. 2007].

are allowed to communicate among themselves in addition to performing the actions of Γ and the payoffs are just as in Γ.

Definition 2 *$\vec{\sigma}$ is a k-*resilient equilibrium *in a game Γ if $\vec{\sigma}$ is a k-resilient$'$ equilibrium in the cheap-talk extension of Γ (where we identify the strategy σ_i in the game Γ with the strategy in the cheap-talk game where player i never sends any messages beyond those sent according to σ_i).* ∎

A standard assumption in game theory is that utilities are (commonly) known; when we are given a game we are also given each player's utility.When players make decision, they can take other players' utilities into account. However, in large systems, it seems almost invariably the case that there will be some fraction of users who do not respond to incentives the way we expect. For example, in a peer-to-peer network like Kazaa or Gnutella, it would seem that no rational agent should share files. Whether or not you can get a file depends only on whether other people share files; on the other hand, it seems that there are disincentives for sharing (the possibility of lawsuits, use of bandwidth, etc.). Nevertheless, people do share files. However, studies of the Gnutella network have shown almost 70 percent of users share no files and nearly 50 percent of responses are from the top 1 percent of sharing hosts [Adar and Huberman 2000].

One reason that people might not respond as we expect is that they have utilities that are different from those we expect. Alternatively, the players may be irrational, or (if moves are made using a computer) they may be playing using a faulty computer and thus not able to make the move they would like, or they may not understand how to get the computer to make the move they would like. Whatever the reason, it seems important to design strategies that tolerate such unanticipated behaviors, so that the payoffs of the users with "standard" utilities do not get affected by the nonstandard players using different strategies. This can be viewed as a way of adding fault tolerance to equilibrium notions.

Definition 3 *A joint strategy $\vec{\sigma}$ is t-*immune *if, for all $T \subseteq N$ with $|T| \leq t$, all joint strategies $\vec{\tau}$, and all $i \notin T$, we have $u_i(\vec{\sigma}_{-T}, \vec{\tau}_T) \geq u_i(\vec{\sigma})$.* ∎

The notion of t-immunity and k-resilience address different concerns. For t immunity, we consider the payoffs of the players not in C; for resilience, we consider the payoffs of players in C. It is natural to combine both notions. Given a game Γ, let $\Gamma^T_{\vec{\tau}}$ be the game that is identical to Γ except that the players in T are fixed to playing strategy $\vec{\tau}$.

Definition 4 *$\vec{\sigma}$ is a (k, t)-*robust *equilibrium if $\vec{\sigma}$ is t-immune and, for all $T \subseteq N$ such that $|T| \leq t$ and all joint strategies $\vec{\tau}$, $\vec{\sigma}_{-T}$ is a k-resilient strategy of $\Gamma^{\vec{\tau}}_T$.* ∎

To state the results of Abraham et al. [2006, 2007] on implementing mediators, three games need to be considered: an *underlying game* Γ, an extension Γ_d of Γ with a mediator, and a cheap-talk extension Γ_{CT} of Γ. Assume that Γ is a *normal-form Bayesian game*: each player has a type from some type space with a known distribution over types, and the utilities of the agents depend on the types and actions taken. Roughly speaking, a cheap talk game *implements* a game with a mediator if it induces the same distribution over actions in the underlying game, for each type vector of the players. With this background, I can summarize the results of Abraham et al. [2006, 2007].

- If $n > 3k + 3t$, a (k, t)-robust strategy $\vec{\sigma}$ with a mediator can be implemented using cheap talk (that is, there is a (k, t)-robust strategy $\vec{\sigma}'$ in a cheap talk game such that $\vec{\sigma}$ and $\vec{\sigma}'$ induce the same distribution over actions in the underlying game). Moreover, the implementation requires no knowledge of other agents' utilities, and the cheap talk protocol has bounded running time that does not depend on the utilities.

- If $n \leq 3k + 3t$ then, in general, mediators cannot be implemented using cheap talk without knowledge of other agents' utilities. Moreover, even if other agents' utilities are known, mediators cannot, in general, be implemented without having a $(k+t)$-punishment strategy (that is, a strategy that, if used by all but at most $k + t$ players, guarantees that every player gets a worse outcome than they do with the equilibrium strategy) nor with bounded running time.

- If $n > 2k + 3t$, then mediators can be implemented using cheap talk if there is a punishment strategy (and utilities are known) in finite expected running time that does not depend on the utilities.

- If $n \leq 2k + 3t$ then mediators cannot, in general, be implemented, even if there is a punishment strategy and utilities are known.

- If $n > 2k + 2t$ and there are broadcast channels then, for all ϵ, mediators can be ϵ-implemented (intuitively, there is an implementation where players get utility within ϵ of what they could get by deviating) using cheap talk, with bounded expected running time that does not depend on the utilities.

- If $n \leq 2k+2t$ then mediators cannot, in general, be ϵ-implemented, even with broadcast channels. Moreover, even assuming cryptography and polynomially-bounded players, the expected running time of an implementation must depend on the utility functions of the players and ϵ.

- If $n > k + 3t$ then, assuming cryptography and polynomially-bounded players, mediators can be ϵ-implemented using cheap talk, but if $n \leq 2k + 2t$, then the running time depends on the utilities in the game and ϵ.

- If $n \leq k + 3t$, then even assuming cryptography, polynomially-bounded players, and a $(k + t)$-punishment strategy, mediators cannot, in general, be ϵ-implemented using cheap talk.

- If $n > k + t$ then, assuming cryptography, polynomially-bounded players, and a public-key infrastructure (PKI), we can ϵ-implement a mediator.

The proof of these results makes heavy use of techniques from computer science. All the possibility results showing that mediators can be implemented use techniques from secure multiparty computation. The results showing that that if $n \leq 3k + 3t$, then we cannot implement a mediator without knowing utilities, even if there is a punishment strategy, uses the fact that Byzantine agreement cannot be reached if $t < n/3$; the impossibility result for $n \leq 2k + 3t$ also uses a variant of Byzantine agreement.

A related line of work considers implementing mediators assuming stronger primitives (which cannot be implemented in computer networks); see [Izmalkov et al. 2005; Lepinski et al. 2004] for details.

6 Other Topics

There are many more areas of interaction between computer science than I have indicated in this brief survey. I briefly mention a few others here:

- *Interactive epistemology:* Since the publication of Aumann's [1976] seminal paper, there has been a great deal of activity in trying to understand the role of knowledge in games, and providing epistemic analyses of solution concepts (see [Battigalli and Bonanno 1999] for a survey). In computer science, there has been a parallel literature applying epistemic logic to reason about distributed computation. One focus of this work has been on characterizing the level of knowledge needed to solve certain problems. For example, to achieve Byzantine agreement common knowledge among the nonfaulty agents of an initial value is necessary and sufficient. More generally, in a precise sense, common knowledge is necessary and sufficient for coordination. Another focus has been on defining logics that capture the reasoning of resource-bounded agents. This work has ranged from logics for reasoning about awareness, a topic that has been explored in both computer science and game theory (see, for example, [Dekel, Lipman, and Rustichini 1998; Fagin and Halpern 1988; Halpern 2001; Halpern and Rêgo 2006; Heifetz, Meier, and Schipper 2006; Modica and Rustichini 1994; Modica and Rustichini 1999]) and logics for capturing *algorithmic knowledge*, an approach that takes seriously the assumption that agents must explicitly compute what they know. See [Fagin et al. 1995] for an overview of the work in epistemic logic in computer science.

- *Network growth:* If we view networks as being built by selfish players (who decide whether or not to build links), what will the resulting network look like? How does the growth of the network affect its functionality? For example, how easily will influence spread through the network? How easy is it to route traffic? See [Fabrikant et al. 2003; Kempe et al. 2003] for some recent computer science work in this burgeoning area.

- *Efficient representation of games:* Game theory has typically focused on "small" games, often 2- or 3-player games, that are easy to describe, such as prisoner's dilemma, in order to understand subtleties regarding basic issues such as rationality. To the extent that game theory is used to tackle larger, more practical problems, it will become important to find efficient techniques for describing and analyzing games. By way of analogy, $2^n - 1$ numbers are needed to describe a probability distribution on a space characterized by n binary random variables. For $n = 100$ (not an unreasonable number in practical situations), it is impossible to write down the probability distribution in the obvious way, let alone do computations with it. The same issues will surely arise in large games. Computer scientists use graphical approaches, such as *Bayesian networks* and *Markov networks* [Pearl 1988], for representing and manipulating probability measures on large spaces. Similar techniques seem applicable to games; see, for example, [Koller and Milch 2001; La Mura 2000; Kearns et al. 2001], and [Kearns 2007] for a recent overview. Note that representation is also an issue when we consider the complexity of problems such as computing Nash or correlated equilibria. The complexity of a problem is a function of the size of the input, and the size of the input (which in this case is a description of the game) depends on how the input is represented.

- *Learning in games:* There has been a great deal of work in both computer science and game theory on learning to play well in different settings (see [Fudenberg and Levine 1998] for an overview of the work in game theory). One line of research in computer science has involved learning to play optimally in a reinforcement learning setting, where an agent interacts with an unknown (but fixed) environment. The agent then faces a fundamental tradeoff between *exploration* and *exploitation*. The question is how long it takes to learn to play well (i.e., to get a reward within some fixed ϵ of optimal); see [Brafman and Tennenholtz 2002; Kearns and Singh 1998] for the current state of the art. A related question is efficiently finding a strategy minimizes *regret*—that is, finding a strategy that is guaranteed to do not much worse than the best strategy would have done in hindsight (that is, even knowing what the opponent would have done). See [Blum and Mansour 2007] for a recent overview of work on this problem.

References

Abraham, I., D. Dolev, R. Gonen, and J. Halpern (2006). Distributed computing meets game theory: Robust mechanisms for rational secret sharing and multiparty computation. In *Proc. 25th ACM Symposium on Principles of Distributed Computing*, pp. 53–62.

Abraham, I., D. Dolev, and J. Halpern (2007). Lower bounds on implementing robust and resilient mediators. Unpublished manuscript.

Adar, E. and B. Huberman (2000). Free riding on Gnutella. *First Monday 5*(10).

Archer, A. and É. Tardos (2001). Truthful mechanisms for one-parameter agents. In *Proc. 42nd IEEE Symposium on Foundations of Computer Science*, pp. 482–491.

Archer, A. and É. Tardos (2002). Frugal path mechanisms. In *Proc. 13th ACM-SIAM Symposium on Discrete Algorithms*, pp. 991–999.

Aumann, R. J. (1959). Acceptable points in general cooperative n-person games. In A. Tucker and R. Luce (Eds.), *Contributions to the Theory of Games IV, Annals of Mathematical Studies 40*, pp. 287–324. Princeton, N. J.: Princeton University Press.

Aumann, R. J. (1976). Agreeing to disagree. *Annals of Statistics 4*(6), 1236–1239.

Aumann, R. J. (1987). Correlated equilibrium as an expression of Bayesian rationality. *Econometrica 55*, 1–18.

Axelrod, R. (1984). *The Evolution of Cooperation*. New York: Basic Books.

Babaoglu, O. (1987). On the reliability of consensus-based fault-tolerant distributed computing systems. *ACM Trans. on Computer Systems 5*, 394–416.

Barany, I. (1992). Fair distribution protocols or how the players replace fortune. *Mathematics of Operations Research 17*, 327–340.

Battigalli, P. and G. Bonanno (1999). Recent results on belief, knowledge and the epistemic foundations of game theory. *Research in Economics 53*(2), 149–225.

Ben-Or, M. (1983). Another advantage of free choice: completely asynchronous agreement protocols. In *Proc. 2nd ACM Symp. on Principles of Distributed Computing*, pp. 27–30.

Ben-Porath, E. (2003). Cheap talk in games with incomplete information. *Journal of Economic Theory 108*(1), 45–71.

Blum, A. and Y. Mansour (2007). Learning, regret minimization, and equilibria. In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani (Eds.), *Algorithmic Game Theory*. Cambridge, U.K.: Cambridge University Press.

Blum, B., C. R. Shelton, and D. Koller (2003). A continuation method for Nash equilibria in structured games. In *Proc. Eighteenth International Joint Conference on Artificial Intelligence (IJCAI '03)*, pp. 757–764.

Brafman, R. I. and M. Tennenholtz (2002). R-MAX: A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research 3*, 213–231.

Chen, X. and X. Deng (2006). Settling the complexity of 2-player Nash equilibrium. In *Proc. 47th IEEE Symposium on Foundations of Computer Science*, pp. 261–272.

Chor, B. and C. Dwork (1989). Randomization in Byzantine agreement. In *Advances in Computing Research 5: Randomness and Computation*, pp. 443–497. JAI Press.

Chu, F. and J. Y. Halpern (2001). A decision-theoretic approach to reliable message delivery. *Distributed Computing 14*, 359–389.

Clarke, E. H. (1971). Multipart pricing of public goods. *Public Choice 11*, 17–33.

Conitzer, V., J. Lang, and T. Sandholm (2007). When are elections with few candidates hard to manipulate? *Journal of the ACM*. To appear.

Conitzer, V. and T. Sandholm (2003). Complexity results about Nash equilibria. In *Proc. Eighteenth International Joint Conference on Artificial Intelligence (IJCAI '03)*, pp. 765–771.

Cramton, P., Y. Shoham, and R. Steinberg (Eds.) (2006). *Combinatorial Auctions*. Cambridge, Mass.: MIT Press.

Daskalis, C., P. W. Goldberg, and C. H. Papadimitriou (2006). The complexity of computing a Nash equilibrium. In *Proc. 38th ACM Symposium on Theory of Computing*, pp. 71–78.

Dekel, E., B. Lipman, and A. Rustichini (1998). Standard state-space models preclude unawareness. *Econometrica 66*, 159–173.

Eliaz, K. (2002). Fault-tolerant implementation. *Review of Economic Studies 69*(3), 589–610.

Fabrikant, A., A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker (2003). On a network creation game. In *Proc. 22nd ACM Symposium on Principles of Distributed Computing*, pp. 347–351.

Fagin, R. and J. Y. Halpern (1988). Belief, awareness, and limited reasoning. *Artificial Intelligence 34*, 39–76.

Fagin, R., J. Y. Halpern, Y. Moses, and M. Y. Vardi (1995). *Reasoning about Knowledge*. Cambridge, Mass.: MIT Press. A revised paperback edition was published in 2003.

Feigenbaum, J., C. Papadimitriou, and S. Shenker (2000). Sharing the cost of muliticast transmissions (preliminary version). In *Proc. 32nd ACM Symposium on Theory of Computing*, pp. 218–227.

Fischer, M. J. (1983). The consensus problem in unreliable distributed systems. In M. Karpinski (Ed.), *Foundations of Computation Theory*, Lecture Notes in Computer Science, Volume 185, pp. 127–140. Berlin/New York: Springer-Verlag.

Fischer, M. J., N. A. Lynch, and M. S. Paterson (1985). Impossibility of distributed consensus with one faulty processor. *Journal of the ACM 32*(2), 374–382.

Fudenberg, D. and D. Levine (1998). *The Theory of Learning in Games*. MIT Press.

Gibbard, A. (1973). Manipulation of voting schemes. *Econometrica 41*, 587–602.

Gilboa, I. and E. Zemel (1989). Nash and correlated equilibrium: some complexity considerations. *Games and Economic Behavior 1*, 80–93.

Goldreich, O., S. Micali, and A. Wigderson (1987). How to play any mental game. In *Proc. 19th ACM Symp. on Theory of Computing*, pp. 218–229.

Govindan, S. and R. Wilson (2003). A global Newton method to compute Nash equilibria. *Journal of Economic Theory 110*(1), 65–86.

Groves, T. (1973). Incentives in teams. *Eocnometrica 41*, 617–631.

Halldórsson, M. M., J. Y. Halpern, L. Li, and V. Mirrokni (2004). On spectrum sharing games. In *Proc. 23rd ACM Symposium on Principles of Distributed Computing*, pp. 107–114.

Halpern, J. Y. (2001). Alternative semantics for unawareness. *Games and Economic Behavior 37*, 321–339.

Halpern, J. Y. (2003). A computer scientist looks at game theory. *Games and Economic Behavior 45*(1), 114–132.

Halpern, J. Y. and L. C. Rêgo (2006). Reasoning about knowledge of unawareness. In *Principles of Knowledge Representation and Reasoning: Proc. Tenth International Conference (KR '06)*, pp. 6–13. Full version available at arxiv.org/cs.LO/0603020.

Halpern, J. Y. and V. Teague (2004). Rational secret sharing and multiparty computation: extended abstract. In *Proc. 36th ACM Symposium on Theory of Computing*, pp. 623–632.

Heifetz, A., M. Meier, and B. Schipper (2006). Interactive unawareness. *Journal of Economic Theory 130*, 78–94.

Heller, Y. (2005). A minority-proof cheap-talk protocol. Unpublished manuscript.

Hopcroft, J. E. and J. D. Ullman (1979). *Introduction to Automata Theory, Languages and Computation*. New York: Addison-Wesley.

Izmalkov, S., S. Micali, and M. Lepinski (2005). Rational secure computation and ideal mechanism design. In *Proc. 46th IEEE Symp. Foundations of Computer Science*, pp. 585–595.

Kearns, M. (2007). Graphical games. In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani (Eds.), *Algorithmic Game Theory*. Cambridge, U.K.: Cambridge University Press.

Kearns, M., M. L. Littman, and S. P. Singh (2001). Graphical models for game theory. In *Proc. Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI 2001)*, pp. 253–260.

Kearns, M. and S. P. Singh (1998). Near-optimal reinforcement learning in polynomial time. In *Proc. 15th International Conference on Machine Learning*, pp. 260–268.

Kearns, M. J. and M. K. Reiter (Eds.) (2005). *ACM Conference on Electronic Commerce (EC '05)*. New York: ACM. See www.informatik.uni-trier.de/ ley/db/conf/sigecom/sigecom2005.html for contents.

Kempe, D., J. Kleinberg, and É. Tardos (2003). Maximizing the spread of influence through a social network. In *Proc. Ninth ACM SIGKDD International Conference Knowledge Discovery and Data Mining*, pp. 137–146.

Kfir-Dahav, N. E., D. Monderer, and M. Tennenholtz (2000). Mechanism design for resource-bounded agents. In *International Conference on Multiagent Systems*, pp. 309–316.

Koller, D. and B. Milch (2001). Structured models for multiagent interactions. In *Theoretical Aspects of Rationality and Knowledge: Proc. Eig hth Conference (TARK 2001)*, pp. 233–248.

Koutsoupias, E. and C. H. Papadimitriou (1999). Worst-case equilibria. In *Proc. 16th Conference on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, Volume 1563, pp. 404–413. Berlin: Springer-Verlag.

Kushilevitz, E. and N. Nisan (1997). *Communication Complexity*. Cambridge, U.K.: Cambridge University Press.

La Mura, P. (2000). Game networks. In *Proc. Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI 2000)*, pp. 335–342.

Lehmann, D., R. Müller, and T. Sandholm (2006). The winner determination problem. In P. Cramton, Y. Shoham, and R. Steinberg (Eds.), *Combinatorial Auctions*. Cambridge, Mass.: MIT Press.

Lemke, C. E. and J. J. T. Howson (1964). Equilibrium poitns of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics 12*, 413–423.

Lepinski, M., S. Micali, C. Peikert, and A. Shelat (2004). Completely fair SFE and coalition-safe cheap talk. In *Proc. 23rd ACM Symp. Principles of Distributed Computing*, pp. 1–10.

Linial, N. (1994). Games computers play: Game-theoretic aspects of computing. In R. J. Aumann and S. Hart (Eds.), *Handbook of Game Theory*, Volume II, pp. 1340–1395. Amsterdam: North-Holland.

Modica, S. and A. Rustichini (1994). Awareness and partitional information structures. *Theory and Decision 37*, 107–124.

Modica, S. and A. Rustichini (1999). Unawareness and partitional information structures. *Games and Economic Behavior 27*(2), 265–298.

Monderer, D. and M. Tennenholtz (1999a). Distributed games. *Games and Economic Behavior 28*, 55–72.

Monderer, D. and M. Tennenholtz (1999b). Distributed Games: From Mechanisms to Protocols. In *Proc. Sixteenth National Conference on Artificial Intelligence (AAAI '99)*, pp. 32–37.

Moreno, D. and J. Wooders (1996). Coalition-proof equilibrium. *Games and Economic Behavior 17*(1), 80–112.

Nash, J. (1950). Equilibrium points in n-person games. *Proc. National Academy of Sciences 36*, 48–49.

Neyman, A. (1985). Bounded complexity justifies cooperation in finitely repated prisoner's dilemma. *Economic Letters 19*, 227–229.

Nisan, N. (2006). Bidding langages for combinatorial auctions. In *Combinatorial Auctions*. Cambridge, Mass.: MIT Press.

Nisan, N. and A. Ronen (2000). Computationally feasible VCG mechanisms. In *Second ACM Conference on Electronic Commerce (EC '00)*, pp. 242–252.

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

Nisan, N., T. Roughgarden, É. Tardos, and V. Vazirani (Eds.) (2007). *Algorithmic Game Theory*. Cambridge, U.K.: Cambridge University Press.

Nisan, N. and I. Segal (2005). Exponential communication inefficiency of demand queries. In *Theoretical Aspects of Rationality and Knowledge: Proc. Tenth Conference (TARK 2005)*, pp. 158–164.

Papadimitriou, C. H. (1994a). *Computational Complexity*. Reading, Mass.: Addison Wesley.

Papadimitriou, C. H. (1994b). On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences 48*(3), 498–532.

Papadimitriou, C. H. (2001). Algorithms, games, and the internet. In *Proc. 33rd ACM Symposium on Theory of Computing*, pp. 749–753.

Papadimitriou, C. H. (2005). Computing correlated equilibria in multiplayer games. In *Proc. 37th ACM Symposium on Theory of Computing*, pp. 49–56.

Papadimitriou, C. H. (2007). The complexity of finding Nash equilibria. In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani (Eds.), *Algorithmic Game Theory*. Cambridge, U.K.: Cambridge University Press.

Papadimitriou, C. H. and T. Roughgarden (2005). Computing equilibria in multi-player games. In *Proc. 16th ACM-SIAM Symposium on Discrete Algorithms*, pp. 82–91.

Papadimitriou, C. H. and M. Yannakakis (1994). On complexity as bounded rationality. In *Proc. 26th ACM Symposium on Theory of Computing*, pp. 726–733.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. San Francisco: Morgan Kaufmann.

Pease, M., R. Shostak, and L. Lamport (1980). Reaching agreement in the presence of faults. *Journal of the ACM 27*(2), 228–234.

Porter, R., E. Nudelman, and Y. Shoham (2004). Simple search methods for finding a Nash equilibrium. In *Proc. Twenty-First National Conference on Artificial Intelligence (AAAI '04)*, pp. 664–669.

Rabin, M. O. (1983). Randomized Byzantine generals. In *Proc. 24th IEEE Symp. on Foundations of Computer Science*, pp. 403–409.

Roughgarden, T. and É. Tardos (2002). How bad is selfish routing? *Journal of the ACM 49*(2), 236–259.

Rubinstein, A. (1986). Finite automata play the repeated prisoner's dilemma. *Journal of Economic Theory 39*, 83–96.

Rubinstein, A. (1998). *Modeling Bounded Rationality*. Cambridge, Mass.: MIT Press.

Sandholm, T. and M. Yakoo (Eds.) (2003). *The Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*. ACM. See http://www.informatik.uni-trier.de/ ley/db/conf/atal/aamas2003.html for the contents.

Satterthwaite, M. (1975). Strategy-proofness and Arrow's conditions: existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory 10*, 187–217.

Shamir, A., R. L. Rivest, and L. Adelman (1981). Mental poker. In D. A. Klarner (Ed.), *The Mathematical Gardner*, pp. 37–43. Boston, Mass.: Prindle, Weber, and Schmidt.

Tennenholtz, M. (2002). Competitive safety analysis: robust decision-making in multi-agent systems. *Journal of A.I. Research 17*, 363–378.

Urbano, A. and J. E. Vila (2002). Computational complexity and communication: Coordination in two-player games. *Econometrica 70*(5), 1893–1927.

Urbano, A. and J. E. Vila (2004). Computationally restricted unmediated talk under incomplete information. *Economic Theory 23*(2), 283–320.

Vetta, A. (2002). Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In *Proc. 43rd IEEE Symposium on Foundations of Computer Science*, pp. 416–425.

Vickrey, W. (1961). Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance 16*, 8–37.

Yao, A. (1982). Protocols for secure computation (extended abstract). In *Proc. 23rd IEEE Symp. on Foundations of Computer Science*, pp. 160–164.