Research Statement (Long Version)

Vincent Conitzer

December 19, 2005

1 Introduction

The Internet allows many parties worldwide to interact with unprecedented amounts of speed, convenience, and formality, and the opportunities that this creates for the increased welfare of humanity are endless. For example, we can create new marketplaces of a size and complexity never before seen in history; we can dramatically increase opportunities for citizens to directly participate in government; and, more generally, we can aggregate the knowledge and preferences of the many to make truly good assessments and decisions in nearly every domain. The parties participating in such mechanisms can be humans, but they can also be software agents that act in their owners’ interests, allowing us to increase the number and complexity of these mechanisms even further.

From a computer science perspective, one unusual aspect of the resulting systems is that the designer has no direct control over one of the system’s core components, namely the behavior of the agents. Generally, these agents will pursue their own self-interest (or, in the case of software agents, the interests of their owners), rather than follow any prescribed behavior; and this self-interested behavior can prevent the realization of the system’s potential benefits. The only solution is to design the mechanism in such a way that we obtain good outcomes in spite of this strategic behavior by the agents. To do so, we need to integrate techniques from economics—specifically, from game theory and mechanism design—into computer science. An additional benefit of doing so is that it allows us to create artificial intelligence with sophisticated strategic reasoning (e.g., computer players for games such as poker). These topics are the focus of my research, and its aim is to enable the societal benefits described above. Below, I will discuss individual research topics, and conclude with my future research plans.

2 Expressive marketplaces

We are witnessing the beginning of a shift in the way marketplaces are designed. Traditional marketplaces have forced their participants to express their preferences in very simplified and approximate fashions. For example, in auctions, bidders have had to place separate bids on individual items, even though how much they valued one item depended on whether they won another item. In contrast, new marketplaces are beginning to allow their participants to express their preferences in their full complexity. This has the potential to greatly improve the efficiency of the final resource allocation, but it comes at the cost of making the market mechanism all the more complex to execute.

2.1 Combinatorial auctions and exchanges

The canonical example of an expressive marketplace is a combinatorial auction, in which there are multiple items for sale, and bidders are allowed to place bids on any subset of the items. While the expressiveness that this provides to the bidders increases economic welfare, several nontrivial issues must be addressed in order for a combinatorial auction to be implemented. First, there is the winner determination problem of deciding which bids to accept to maximize the total value. This problem is known to be NP-hard (even to approximate). Second, there is the elicitation problem of obtaining enough information from the bidders about their valuation functions to solve the winner determination problem, while asking only a reasonable number of queries.
The winner determination problem can become easier to solve if the bids display some special structure. We studied the following structure: the items are the vertices of a graph, and every bid is on a connected component of the graph. We showed that if the treewidth of the graph is bounded, then the winner determination problem can be solved in polynomial time. Moreover, we showed that if the graph is a tree, then we can construct the graph from the bids, so that we do not need to know the graph beforehand. (However, in general, we proved that minimizing the number of edges in the graph is NP-hard.) We also showed that the winner determination and graph construction problems become NP-hard when bids are on a few components (rather than a single one) [3]. As another example of a special structure that the bids may have, we studied the setting in which each bidder’s valuation has limited interdependency among the items. We showed that, unfortunately, the winner determination problem remains NP-hard even with the minimum degree of interdependency, unless there is additional structure. However, the bidders’ valuations are much easier to elicit when the degree of interdependency is small [32]. We also showed more generally that what classes of valuations can be efficiently elicited by querying the bidders only for the values of specific bundles [35].

2.2 Public goods and externalities

While most of the research done on expressive marketplaces so far has focused on combinatorial auctions and exchanges, I believe that there are myriad other applications of this framework that are waiting to be uncovered. One domain that I feel is particularly important in this regard is that of public goods. Rather than just benefit a single owner, public goods can benefit large groups of agents. Examples include clean air, defense capabilities, pure research, etc. Public goods are closely related to the notion of an externality. An externality takes place when actions by some agents affect the utilities of otherwise uninvolved agents. It can be argued that many of the world’s major problems today are caused by inefficient solutions to public goods/externalities problems.

One specific setting that we have studied is that of negotiating donations to charitable causes. When donating money to a charity, it is possible to use the contemplated donation as negotiating material to induce other parties interested in the charity to donate more. Such negotiation is sometimes done through simple matching offers, but these allow for only limited negotiation. We introduced a language for expressing very general types of matching offers over multiple charities, and formulated the corresponding clearing problem (deciding how much each bidder pays, and how much each charity receives). While we showed that this problem is in general NP-hard to approximate to any ratio, we also provided a mixed-integer programming based approach for solving it optimally. Moreover, we showed that the problem can be solved optimally in polynomial time in certain special cases (though it remains hard in others) [22]. At a 2005 workshop in Dagstuhl, I used this approach to raise money for tsunami victims.

We have also introduced a more general framework for negotiation in settings with externalities. Here, we showed that finding a nontrivial feasible solution is NP-hard even in most special cases, but for one important setting (where each agent controls only a single variable and all externalities are negative), we provided an efficient algorithm for finding the feasible solution that maximizes the concessions made by each agent (maximizing the sum of utilities, also known as social welfare, remains NP-hard). We also identified a special case in which social welfare can be maximized in polynomial time using linear programming [28].

3 Social choice theory (voting)

Voting (or rank aggregation) is a general method for aggregating multiple agents’ preferences over a set of alternatives. A voting rule takes as input every voter’s vote (typically, a ranking of the alternatives), and produces as output either just the winning alternative, or an aggregate ranking of the alternatives. The alternatives can be potential representatives, but also plans, allocations of resources, etc. Rank aggregation can also be useful when the rankings do not represent a conflict of interest among voters, but rather a disagreement of judgement. When multiple entities try to assess some absolute quality of the alternatives, we may wish to turn their individual rankings into an aggregate ranking. Examples include judges ranking participants, search engines ranking webpages, etc.
There are many different voting rules, which will sometimes give different outcomes for the same votes. For example, we can rank alternatives simply by how often they are ranked first in a vote (the plurality rule); or, we can give each alternative a point for each time that a voter ranks it higher than some other alternative, and rank candidates according to their total scores (the Borda rule). Which voting rule should be used depends on a variety of factors, some of which are discussed below.

3.1 Hard-to-execute voting rules

Not all voting rules are computationally easy to execute. Two important voting rules that are NP-hard to execute are the Kemeny rule, which minimizes the number of times that the aggregate ranking disagrees with a vote on the order of a pair of candidates; and the Slater rule, which minimizes the number of times that the aggregate ranking disagrees with the majority of votes on the order of a pair of candidates. One can try to approximate these criteria, but this is effectively changing the voting rule. One can also turn to (worst-case exponential time) search-based approaches. We derived a number of upper bounding techniques to use in search, based on cycles in the pairwise majority graph and linear programming relaxations [2]. I also discovered a highly effective preprocessing technique for computing Slater rankings, which relies on identifying sets of similar candidates. I used this concept of a set of similar candidates to give the first “direct” proof that Slater rankings are NP-hard to compute even when there are no pairwise ties [1]. (The only previous proof was a derandomization of a randomized reduction, both given in 2005; the conjecture had been open for thirteen years before that.)

3.2 Voting rules that are hard to manipulate

An important problem in social choice is that of manipulation, that is, misreporting one’s preferences in order to obtain a better outcome for oneself. The famous Gibbard-Satterthwaite theorem states that when there are three or more alternatives, essentially all voting rules are manipulable (that is, there are situations in which voters would be better off misreporting their preferences). Fortunately, this need not be a problem if it is computationally hard for voters to discover the potential for beneficial manipulation. Prior work had already identified two voting rules that are NP-hard to manipulate, but most rules are in fact easy to manipulate.

We studied whether it is possible to modify voting rules to make them hard to manipulate. Specifically, we studied adding a preround in which pairs of alternatives face off in pairwise elections to determine who continues to the final round; this final round is then executed according to the original rule. Surprisingly, this is enough to make almost any rule hard to manipulate! Specifically, any rule satisfying a certain technical condition (which includes most common rules) becomes NP-hard, #P-hard, or PSPACE-hard to manipulate, depending on whether the schedule of the preround is determined before the votes are collected, after the votes are collected, or the scheduling and the vote collecting are interleaved, respectively [17]. These are the first results where manipulation is more than NP-hard (assuming $\text{NP} \neq \text{PSPACE}$).

All of these hardness results rely on the number of alternatives being unbounded. This is a reasonable model when voting over (say) allocations of tasks or resources, but in many settings, the number of alternatives is small. Thus, we asked whether manipulation could still be hard when the number of alternatives is a small constant. We showed that coalitional manipulation by weighted voters is indeed NP-hard even with a constant number of alternatives for most (but not all) common voting rules. We also showed that when this type of manipulation is hard, then manipulation by an unweighted individual is also hard if that individual is uncertain about the others’ votes. The smallest number of alternatives at which a rule becomes hard to manipulate depends on the rule as well as on the type of manipulation (e.g., making a specific alternative lose is easier than making a specific alternative win) [8, 4].

3.3 Eliciting votes efficiently

As in the setting of combinatorial auctions and exchanges, we are faced with the elicitation problem of eliciting enough information about the voters’ preferences to determine the winner. For the common voting rules, we studied the computational complexity of deciding whether enough information has been elicited to determine a
winner, as well as how to elicit the votes most efficiently given suspicions on how the voters will vote. In the
same work, we also showed that elicitation may introduce additional opportunities for manipulation, but we also
showed how to prevent this by restricting attention to a smaller class of elicitation protocols [10]. We also studied
the communication complexity of the common voting rules. We derived lower bounds on the worst-case number
of bits that need to be elicited, and constructively derived matching upper bounds by giving efficient elicitation
protocols [26].

3.4 Interpreting voting rules as maximum likelihood estimators

One potential view of voting is the following. There exists a “correct” outcome (winner/ranking), and each voter’s
vote corresponds to a noisy perception of this correct outcome. If we are given the noise model, then for any vector
of votes, we can compute the maximum likelihood estimate of the correct outcome. This maximum likelihood
estimate constitutes a voting rule. We studied for which common voting rules there exists a noise model such
that the rule is the maximum likelihood estimate for that noise model. It turns out that only some of the common
voting rules have this property, and which ones they are depends on whether the outcome is a winner or a complete
ranking [25].

4 Mechanism design

Aggregating the preferences of self-interested agents is complicated by the fact that they will misreport their
preferences if this is to their benefit. Mechanism design is the study of how to aggregate preferences in such
a way that desirable outcomes are obtained in spite of such strategic behavior. According to a result known as
the revelation principle, it suffices to consider the following approach: have the agents report their preferences
directly, and choose outcomes in such a way that no agent has an incentive to misreport. (This is called a truthful
mechanism.) Classical mechanism design has produced several general-purpose truthful mechanisms. The best-
known such mechanism is the VCG (or Clarke tax) mechanism, which requires an agent to pay the externality
that its presence imposes on the other agents. Unfortunately, general mechanisms such as the VCG mechanism
invariably come with certain drawbacks. For example, the VCG mechanism is not budget-balanced: the agents’
payments do not necessarily sum to 0, so that some of the payments must be collected by an outside party (or
burned). If the budget must be balanced, it is in general impossible to create a truthful mechanism that always
chooses the efficient outcome (for example, in the charitable donation setting discussed above, this is impossible).

4.1 Failures of the VCG mechanism in combinatorial auctions and exchanges

Two related problems with the VCG mechanism are that it does not provide strong revenue guarantees, and that
it is vulnerable to collusion. We studied the extent to which these problems can occur in combinatorial auctions
and exchanges. We studied four settings: combinatorial (forward) auctions with free disposal (“free disposal”
means that items can be thrown away at no cost), combinatorial reverse auctions with free disposal (in a reverse
auction, the auctioneer procures items from the bidders), combinatorial forward (or reverse) auctions without free
disposal, and combinatorial exchanges. In each setting, we gave an example of how additional bidders (colluders)
can make the outcome much worse (less revenue or higher cost) under the VCG mechanism (but not under a
first price mechanism); derived necessary and sufficient conditions for such an effective collusion to be possible
under the VCG mechanism; and (when nontrivial) studied the computational complexity of deciding whether these
conditions hold [23].

4.2 Automated mechanism design

General-purpose mechanisms such as the VCG mechanism constitute some of the great achievements of theoretical
economics. Unfortunately, these mechanisms invariably come with a variety of drawbacks. For example, the VCG
mechanism cannot be applied in settings where it is impossible to make payments, or where payments can only be
made from agent to agent (and not to a third party such as an auctioneer); and if our objective is not to obtain the most efficient allocation of resources, but rather (say) to maximize revenue, the VCG mechanism is suboptimal. Other general-purpose mechanisms have their own limitations, and there are various impossibility results (such as the Gibbard-Satterthwaite and Myerson-Satterthwaite theorems) that show that any mechanism will fail in some settings.

In contrast to the classical approach of designing general-purpose mechanisms by hand, we proposed the automated mechanism design approach. Under this approach, optimization algorithms automatically create the mechanism that is optimal for the specific setting and objective at hand. This approach can outperform general-purpose mechanisms (leading to better outcomes or stronger nonmanipulability guarantees), and it can also create mechanisms for settings for which no mechanisms have yet been created.

The drawback of automated mechanism design is that it requires one to solve nontrivial computational problems. We showed that in most typical cases, it is NP-hard to design optimal deterministic mechanisms, but optimal randomized mechanisms (in which the mechanism specifies a distribution over outcomes) can be designed in polynomial time using linear programming. (Turning these linear programs into mixed-integer programs gives an algorithm for designing optimal deterministic mechanisms.) We derived similar results under a more concise representation where bidders’ utilities can decompose over issues [5]. We also developed a fast special-purpose algorithm for a special case [18], and we used our software to obtain solutions for various applications (including examples interesting from an academic viewpoint [11] as well as a few industrial projects). More recently, we have developed techniques for automatically developing multistage mechanisms that can reduce the amount of preference elicitation required [33].

4.3 Designing mechanisms for computationally bounded agents

Mechanism design has traditionally taken the conservative view that agents will always choose the action that is in their own best interest—the assumption of perfect rationality. The revelation principle (discussed earlier) relies heavily on this assumption. However, this assumption may be overly conservative, in that agents may not always have the computational resources to find the action that is in their own best interest—that is, their rationality is bounded. For instance, we have already seen that manipulation can be hard in voting settings.

We have shown that if agents are computationally bounded, there can be significant benefits to using nontruthful mechanisms. First, we studied a setting in which the optimal truthful mechanism is NP-hard to execute for the center. In that setting we show that by moving to nontruthful mechanisms, one can shift the burden of having to solve the NP-hard problem from the center to one of the agents. Second, we studied a new oracle model that captures the setting where utility values can be hard to compute even when all the pertinent information is available—a situation that occurs in many practical applications. In this model we showed that by moving to nontruthful mechanisms, one can shift the burden of having to ask the oracle an exponential number of costly queries from the center to one of the agents. In both cases, the nontruthful mechanism is equally good as the optimal truthful mechanism in the presence of unlimited computation. More interestingly, whereas being unable to carry out either difficult task would have hurt the center in achieving its objective in the truthful setting, if the agent is unable to carry out either difficult task, the value of the center’s objective strictly improves [20].

To be able to take advantage of the benefits of nontruthful mechanisms, we have outlined a new approach to mechanism design. Under this approach, the designer starts with a nontruthful mechanism, and incrementally tries to make the mechanism more truthful over a series of iterations. We exhibited examples in which this approach generates desirable mechanisms, gave algorithms for executing the approach automatically, and showed that the resulting mechanisms are computationally hard to manipulate [7].

\footnote{Specifically, these cases include maximizing social welfare without using payments [9], maximizing arbitrary objectives using payments [12], maximizing the designer’s utility function for the outcome without using payments, and maximizing revenue [24]. We also showed that this last problem is effectively the same as that of computing revenue-maximizing combinatorial auctions when bidders have unit demand, so that problem has the same complexity.}
5 Computing game-theoretic solution concepts

Game theory studies how agents should play when their utilities are affected by the actions of other agents, who pursue their own interests. Such a setting is known as a *game*. The optimal action to take depends on the actions of the other agents, so that in general there is no clear *ex ante* optimal strategy. Rather, game theory relies mostly on equilibrium notions, where the agents are in equilibrium when no agent (or sometimes even no group of agents) has an incentive to deviate from a prescribed strategy. These equilibrium notions (as well as other methods for solving games, where they are available) are known as solution concepts.

Many solution concepts have been proposed, but the study of how to compute strategies according to these solution concepts has received only limited interest until recently. Knowing how to compute game-theoretic strategies has obvious applications to the design of software agents (including, for example, trading agents, RoboSoccer, computer poker players, sensor networks, ...). However, there are other applications as well. For example, if we discover that some equilibria are easier to compute than others, this may help resolve the well-known equilibrium selection problem (if we do not know which equilibrium should be played, then we still do not know how to play). Moreover, there are implications for mechanism design: if the solution concepts are too difficult to compute, then it is unreasonable to expect that agents will play according to them. This would once again invalidate the revelation principle, and we would require another way of evaluating how agents are likely to play in any given (nontruthful) mechanism.

5.1 Noncooperative game theory

The best-known solution concept in game theory is that of Nash equilibrium. A Nash equilibrium assigns a strategy to every player, in such a way that no single player has an incentive to change strategies if the other players play their assigned strategies. In general, we may consider mixed strategies—that is, probability distributions over each player’s “pure” strategies. It is known that every finite game has at least one mixed-strategy Nash equilibrium, but it is not clear whether such an equilibrium can be efficiently computed. Christos Papadimitriou has called this “a most fundamental computational problem whose complexity is wide open” and “together with factoring, [...] the most important concrete open question on the boundary of $P$ today.” (A recent sequence of papers does show that the problem of constructing a Nash equilibrium is complete for a class of problems called PPAD, even when there are only two players.)

We have shown that many related questions are hard [15]. We gave a single reduction that proves that 1) it is NP-hard to determine whether equilibria with certain properties exist (for example, an equilibrium in which a certain pure strategy is sometimes/never played); 2) it is NP-hard to find a Nash equilibrium that maximizes certain objectives (for example, social welfare), and in fact these objectives are inapproximable; 3) it is #P-hard to count the number of Nash equilibria (or connected components of Nash equilibria). We also showed that computing a Nash equilibrium in symmetric games is no easier than in general games; that in Bayesian games, determining whether a pure-strategy Bayes-Nash equilibrium exists is NP-hard; and that in stochastic (Markov) games, determining whether a pure-strategy Nash equilibrium exists is PSPACE-hard even in “invisible” games (and that this remains NP-hard even the game is finite). (This paper is on the reading list of most courses in computational game theory, and its main proof has been rewritten and translated multiple times.)

Multiple (worst-case exponential time) algorithms for computing a Nash equilibrium have been proposed. We have proposed an algorithm for computing Nash equilibria that is based on mixed-integer programming (this approach has the advantage that it can also find optimal equilibria) [34], as well as a nontrivial preprocessing technique that can significantly reduce the sizes of the games to be solved, and that is sufficient to completely solve games in a special class of games, in linear time [30].

Nash equilibrium is not the only solution concept of interest, though. A more basic solution concept is that of dominance. One strategy is said to strictly (weakly) dominate another if it performs strictly better (at least as well and in at least one case strictly better) against any opponent strategies. In iterated dominance, dominated strategies are removed from the game, after which new strategies may become dominated, etc. We studied the complexity of using these solution concepts [27], and showed that all of them can be applied in polynomial time—with the exception of iterated weak dominance, which is NP-complete to use even in 0-1 games (whether or not
dominance by mixed strategies is allowed). We also characterized the complexities of dominance and iterated dominance when dominating strategies are allowed to randomize over only a few pure strategies, as well as in Bayesian games (whose representation is more compact than normal-form games).

Sometimes it is not possible to solve a game using (iterated) dominance. In such cases, do we always need to resort to Nash equilibrium (and deal with the computational complexity that it entails)? The answer is no: we have defined a spectrum of solution concepts (specifically, strategy eliminability criteria), with dominance and Nash equilibrium at opposite ends of the spectrum. In general, it is coNP-complete to compute whether a strategy is eliminable according to these criteria, but it can be done in polynomial time when we are close to the “dominance” end of the spectrum. Additionally, the closer we are to the “dominance” end of the spectrum, the stronger is the justification for eliminating the strategy [29].

We have also investigated how to compute optimal strategies in settings in which one player can commit to a (possibly mixed) strategy first. We showed that optimal strategies can be computed in polynomial time for two-player general-sum games in the case of commitment to mixed strategies (this is a generalization of the well-known polynomial-time computability of minimax strategies), as well as for n-player general-sum games in the case of commitment to pure strategies only. The problem becomes NP-hard in Bayesian games, as well as in games of 3 or more players with commitment to mixed strategies [6].

Of course, for a specific game, it may be much easier to compute an optimal strategy. I recently designed an optimal computer player for (one of the variants of) the game of Liar’s Dice.

5.2 Cooperative/coalitional game theory

How should groups of agents work together, and how should they distribute the gains from cooperation? This is what is studied in cooperative (or coalitional) game theory. Solution concepts in cooperative game theory are concerned with stability as well as fairness. For example, the core is the set of all outcomes from which no coalition has an incentive to deviate; the Shapley value rewards every agent according to how much that agent contributed (in some sense). We studied how to compute these solution concepts under two natural representations of cooperative games [31, 21]. We also showed that in anonymous settings such as the Internet, these concepts are vulnerable to various manipulations, such as the use of multiple pseudonyms by a single agent. We introduced and characterized a stronger concept called the anonymity-proof core which is robust to such manipulations [36].

6 Learning in games

It is not always possible for the players of a game to immediately play according to a solution concept: often the players need to learn how to play, and will only eventually converge to a solution. There are various reasons why learning may be necessary: the players may not know all of the payoffs (or other variables) in the game; the players may not be sophisticated enough to compute the solution concept right away; or multiple outcomes may be consistent with the solution concept, and the players need to coordinate. Moreover, it is possible that the opponent players are not perfectly rational, in which case the player needs to learn how to play well against them.

We gave the first algorithm for learning in games that, in general repeated games, 1) learns to play optimally against (eventually) stationary opponents, and 2) converges to a Nash equilibrium in self-play [13]. (The best previous algorithm achieved these properties only in two-player, two-action games, and required additional assumptions.) In another line of research, we introduced a general theoretical framework for analyzing the cost of having to learn which zero-sum game is being played (when the opponent already knows the game) [14]. In the setting where the opponent’s payoffs are not known, we showed how to give lower bounds on the convergence time of any learning algorithm using the communication complexity of the target solution concept, and we exactly characterized the communication complexity of Nash equilibrium, iterated dominance (strict and weak), and backwards induction [19].
7 Metareasoning

Even in single-agent settings, it is not always feasible to perform all potentially useful deliberation and information gathering actions before making a decision. In such cases, we are confronted with the metareasoning problem of selecting which of these actions to take. We defined three very basic metareasoning problems, and we showed that two of them are NP-hard, and the third even PSPACE-hard [16].

8 Future research

We have only begun to see how techniques from computer science (and artificial intelligence in particular) can be used to drastically improve the efficiency of marketplaces. As a consultant for CombineNet, Inc., a company that focuses mainly on providing optimal solutions for strategic sourcing, I am continuously on the forefront of the integration of these techniques into industry, and it never ceases to amaze me how much potential is yet left untapped. CombineNet has been very successful at creating several types of expressive marketplace, enabled by powerful languages and algorithms. Nevertheless, most negotiation over the allocation of tasks and resources in the world today is still done without the tremendous benefits of expressiveness. Moreover, there are benefits to be obtained beyond those realized by adding expressiveness to the marketplace. For instance, the (potentially automated) design of better market mechanisms will allow us to not only improve what the participants in the marketplace can express, but also what they strategically will express. Indeed, I have helped CombineNet develop techniques and software for automatically designing the rules of the marketplace, and some customers have begun to make use of them.

From a research perspective, even when we only consider the design of new, expressive marketplaces (without any consideration of mechanism design techniques), I see many open questions and opportunities. Some of these concern more “traditional” applications that deal with the allocation of tasks and resources, but I am especially interested in expressive negotiation over public goods (goods which benefit large populations) and, closely related, externalities (effects that one agent’s actions have on an otherwise unrelated agent). Arguably, most of the major problems facing the world today concern externalities and the provision of public goods. Environmental problems such as global warming, toxic chemicals in the environment, deforestation, and overfishing are perhaps some of the clearest examples. Other global issues that can serve as examples include funding basic research and education; detecting and preventing disease outbreaks; controlling international terrorism and illicit drug trade; etc. There are currently no good mechanisms for efficiently and directly aggregating people’s preferences in such domains; rather, we are forced to rely on the political system and a few nongovernmental organizations. To see how limited the expressiveness of these systems is, it suffices to consider how rarely we cast a vote on anything, and what we can vote for. I believe that our initial papers have only scratched the surface of what can be done in terms of directly aggregating preferences over complex public good decisions, and I will seek to address these issues much deeper. Related to this, I will pursue my work on computational aspects of social choice theory. This will facilitate holding more elections over more complex issues, in a way that is less prone to manipulation.

Once we take into account the fact that agents will report their preferences strategically, we are confronted with many more unresolved research questions that I plan to address. Many of the solutions that traditional mechanism design provides become impractical in expressive, computationally nontrivial settings: for example, standard mechanisms may become too computationally expensive to execute. Similarly, some of the traditional impossibility results may not hold up in these settings: for example, computational hardness can provide barriers to strategic misreporting of preferences. There are at least two major directions that I wish to pursue on mechanism design. One is automated mechanism design. This line of research is still in its infancy (we only introduced it in 2002), and there are many open questions here that I want to address. (My advisor recently received a US$1,100,000 NSF ITR grant to continue research on automated mechanism design.) For example, each specific class of problems for which we wish to automatically design mechanisms has its own special structure, and this can potentially be exploited to compute the optimal mechanism faster (and improving the scalability of automated mechanism design is a topic of major importance). Another interesting topic is to make the generated mechanisms easier to understand for humans. The other major direction is the design of mechanisms that exploit agents’
bounded rationality. Our existing research already has identified various isolated situations in which computational hardness can provide a barrier to manipulation, but a general framework for exploiting bounded rationality (to improve the welfare of all) is still severely lacking. I intend to create such a framework.

To best exploit agents’ bounded rationality in the mechanism design process, we must understand how they will behave in mechanisms in which telling the truth is not necessarily optimal. Hence, we need to understand how they play in arbitrary strategic games. Will they play according to standard game-theoretic equilibrium notions? They may, if finding such equilibria is within their computational reach. Thus, the computational complexity of computing the various types of equilibrium in various settings becomes a key question. Of course, designing good algorithms for computing game-theoretic equilibria is a topic of tremendous importance in its own right, especially for AI purposes—for example, for creating an optimal computer player for poker (or Liar’s Dice, see above). Personally, I am also very curious to find out if (informally stated) some equilibria are easier to compute than others. For example, our paper on Nash equilibria complexity results shows that it is hard to compute the optimal equilibrium (for various notions of “optimal”), but computing a (any) equilibrium may be significantly easier. Is it perhaps the case that there is some refinement of Nash equilibrium, such that one such refined equilibrium always exists, and it is easy to find? This could also help resolve the “equilibrium selection problem”—if a game has multiple equilibria, which one should/will be played?—that economists have struggled with for decades. In any case, many open questions remain on the computation of game-theoretic solution concepts, and I intend to pursue these questions further.

Finally, during my years in graduate school, I have been able to create a number of new, unconventional, and important lines of research within my area. (I would argue that some of my most original ideas include the application of expressive negotiation to public goods/externalities settings; the basic computational problem that led to the field of automated mechanism design; techniques for exploiting computational hardness of manipulation in the design of mechanisms; the new strategy eliminability concept; and the use of communication complexity as a lower bound for learning in games.) I hope to be fortunate enough to continue to generate highly creative research ideas.

References


