Designing Auctions for Deliberative Agents

Kate Larson and Tuomas Sandholm

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
{klarson,sandholm}@cs.cmu.edu

Abstract. In many settings, bidding agents for auctions do not know their preferences *a priori*. Instead, they must actively determine them through deliberation (e.g., information processing or information gathering). Agents are faced not only with the problem of deciding how to reveal their preferences to the mechanism but also how to deliberate in order to determine their preferences. For such settings, we have introduced the deliberation equilibrium as the game-theoretic solution concept where the agents' deliberation actions are modeled as part of their strategies. In this paper, we lay out mechanism design principles for such deliberative agents.

We also derive the first impossibility results for such settings - specifically for private-value auctions where the agents' utility functions are quasilinear, but the agents can only determine their valuations through deliberation. We propose a set of intuitive properties which are desirable in mechanisms used among deliberative agents. First, mechanisms should be non-deliberative: the mechanism should not be solving the deliberation problems for the agents. Secondly, mechanisms should be deliberation-proof: agents should not deliberate on others' valuations in equilibrium. Third, the mechanism should be non-deceiving: agents do not strategically misrepresent. Finally, the mechanism should be sensitive: the agents' actions should affect the outcome. We show that no direct-revelation mechanism satisfies these four properties. Moving beyond direct-revelation mechanisms, we show that no value-based mechanism (that is, mechanism where the agents are only asked to report valuations - either partially or fully determined ones) satisfies these four properties.

1 Introduction

Game theory, and mechanism design in particular, have long been successfully used in economics and have recently drawn a lot of research interest from computer scientists (e.g., [7] [8]). In most of this work it is assumed that the participants, or agents, know their preferences and the goal of the mechanism is to extract this information to a sufficient extent, and select an outcome such that desirable properties are achieved. However, there are many settings where agents do not know their preferences *a priori*. Instead they may, for example, have to solve computationally complex optimization problems, query databases, or perform complicated searches in order to determine the worth of an outcome. We call the actions taken to determine preferences *deliberation*.

If there are no restrictions placed on the deliberation capabilities of agents, they could optimally determine their preferences and act as fully rational agents. However,

in many settings there are costs associated with deliberation. Agents are not able to optimally determine their preferences, but instead must trade off quality of valuations against deliberation cost. Decision making under costly deliberation resources is challenging even in single-agent settings. Having to interact with other agents complicates the problem further. Agents must take into account the other agents' actions in determining both how to act in the mechanism and also how to use deliberation resources.

We have proposed explicitly including the deliberation actions of agents into their strategies, and then analyzing games for *deliberation equilibria* which are fixed points in the space of strategy profiles from this enlarged strategy space [4] [5] . Using this approach, we have studied common auction mechanisms such as the first-price auction, Vickrey auction, ascending auction, descending auction, and generalized Vickrey auction. We discovered the existence of interesting strategic behavior. In each auction mechanism studied, there existed instances where, in equilibrium, agents would use their deliberation resources to determine *other agents' valuations* of the item(s) being auctioned. We coined this phenomenon *strategic deliberation*.

In this paper, we build on this body of work. Instead of looking at the properties of auctions that were designed for fully rational agents, we ask the question: "Is it possible to *design* auctions that have desirable properties for such deliberative agents?" We propose a set of weak, intuitive properties that are desirable for auctions designed for such agents. In particular, we propose that auctions should not solve the valuation problems for the agents, that strategic deliberation should not occur in equilibrium, that agents should not have incentive to misreport, and that the agents' actions affect the outcome. We show that no direct-revelation mechanism satisfies these four properties. Moving beyond direct-revelation mechanisms, we show that no value-based mechanism (that is, mechanism where the agents are only asked to report valuations - either partially or fully determined ones) satisfies these four properties.

The rest of the paper is as follows. We first provide an example application where our approach is needed (Section 2). We then given an overview of pertinent mechanism design concepts, and describe the model for computationally-limited agents (Sections 3 and 4). We show that there is a parallel to the revelation principle for our setting, but argue that the direct mechanism produced has highly impractical properties. We then propose a set of auction properties which we believe are important when the auction is to be used among deliberative agents. Our main results show that it is impossible to design a direct-revelation mechanism that satisfies those desirable properties, and furthermore, it is impossible to design any value-based mechanism that satisfies them (Section 5).

2 An Example Application

To make the presentation more concrete, we now discuss an example domain where our methods are needed.

Consider the example presented in Figure 1. An agent is trying to determine how much it values a specific product. In order to help determine its value, the agent queries a product review database to gather information about the product. Each query costs some fixed an amount, so an agent is faced with the decision of how much information

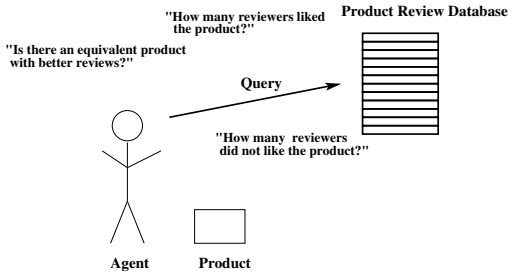

Fig. 1. *Agents may need to gather information in order to determine their value for a certain item. In this figure, an agent gathers information about a product by querying a product review database. Each query returns an answer which is used by the agent to update its value for the product. However, each query costs some fixed amount, and so the agent must decide how much information it needs about the product, given the cost of gathering it.*

to gather given the cost to acquire it. It may also be in the agent's best interest to use some of its deliberation resources (i.e. money to pay for queries) to (partially) determine the values that the other agents in the auction have for the product. By doing some initial deliberating on a competitor's problem, an agent can gather information that may be useful in formulating its own deliberating and bidding strategies.

3 Mechanism Design for Rational Agents

In this section we present an overview of pertinent mechanism design concepts. We assume that the reader has a basic background in game theory. The mechanism design problem is to implement an optimal system-wide solution to a decentralized optimization problem with self-interested agents with private information about their preferences for different outcomes. One of the most exciting applications of mechanism design has been in the area of auction design.

We assume that there is a set of agents, I, $|I| = n$. Each agent, i, has a *type*, $\theta_i \in \Theta_i$, which represents the private information of the agent that is relevant to the agent's decision making. In particular, an agent's type determines its preferences over different outcomes. We use the notation $u_i(o, \theta_i)$ to denote the utility of agent i with type θ_i for outcome $o \in \mathcal{O}$ ($\mathcal{O}$ is the space of possible outcomes). As mentioned in the first paragraph, the goal of mechanism design is to implement some system-wide solution. This is defined in terms of a *social choice function*.

Definition 1 (Social Choice Function). *A* social choice function *is a function $f : \Theta_1 \times \ldots \times \Theta_n \mapsto \mathcal{O}$, such that, for each possible profile of agents' types $\theta = (\theta_1, \ldots, \theta_n)$ assigns an outcome $f(\theta) \in \mathcal{O}$.*

The mechanism design problem is to implement a set of "rules" so that the solution to the social choice function is implemented despite agents' acting in their own self-interest.

Definition 2 (Mechanism). *A mechanism $M = (S_1, \ldots, S_n, g(\cdot))$ defines the set of strategies S_i available to each agent and an outcome rule $g : S_1 \times \ldots \times S_n :\mapsto \mathcal{O}$, such that $g(s)$ is the outcome implemented by the mechanism for strategy profile $s = (s_1, \ldots, s_n)$.*

A mechanism *implements* a social choice function $f(\cdot)$ if there is an equilibrium of the game induced by the mechanism which results in the same outcomes as $f(\cdot)$ for every profile of types, θ.

Definition 3 (Implementation). *A mechanism $M = (S_1, \ldots, S_n, g(\cdot))$* implements *social choice function $f(\cdot)$ if there is an equilibrium strategy profile $s^* = (s_1^*, \ldots, s_n^*)$ such that $g(s^*(\theta)) = f(\theta)$ for all θ.*

An important class of mechanisms are *direct revelation mechanisms*.

Definition 4 (Direct revelation mechanism). *A* direct revelation mechanism *is a mechanism in which $s_i = \Theta_i$ for all i and has outcomes rule $g(\hat{\theta})$ based on reported types $\hat{\theta} = (\hat{\theta_1}, \ldots, \hat{\theta_n})$.*

One of the most important results in mechanism design is the *Revelation Principle*. It states that any mechanism can be transformed into an equivalent direct mechanism where, in equilibrium, all agents truthfully reveal their types (that is, the mechanism is *incentive compatible*).

Theorem 1 (Revelation Principle). *Suppose there exists a mechanism M that implements the social choice function $f(\cdot)$ in dominant strategies (Bayesian-Nash equilibrium). Then $f(\cdot)$ is truthfully implementable in a dominant strategies (Bayesian-Nash) incentive compatible direct-revelation mechanism.*

The Revelation Principle suggests that mechanism designers need only be concerned with direct-revelation mechanisms. Later in the paper we will discuss the Revelation Principle in more detail.

In this paper we restrict ourselves to settings where agents have *quasilinear preferences*. That is, the utility function of an agent i has the form $u_i(o, \theta_i) = v_i(x, \theta_i) + p_i$, where outcome o defines an allocation x and a transfer p_i for the agent.

In general, mechanisms for quasilinear preferences take a certain form.

Definition 5 (Mechanisms for quasilinear environments). *A mechanism for quasilinear environments is a mechanism $M = (S_1, \ldots, S_n, (k(\cdot), t_1(\cdot), \ldots, t_n(\cdot)))$ such that the outcome function $g(\cdot) = (k(\cdot), p_1(\cdot), \ldots, p_n(\cdot))$ where $k : S_1 \times \ldots \times S_n :\mapsto \mathcal{K}$ is a choice rule which selects some choice from choice set $\mathcal{K}$, and transfer rules $p_i : S_1 \times \ldots \times S_n :\mapsto \mathbb{R}$. one for each agent, compute the payment $t_i(s)$ made by agent i.*

In addition to quasilinear environments, we also assume that agents' have private values. This means that an agent's utility depends only on its own type. Many ecommerce applications are set in the quasilinear private-value environment. For example, many auctions belong to this set of mechanisms. The choice rule $k(\cdot)$ specifies which agents are allocated which items, and the transfers, $t_i(\cdot)$, specify the amount each agent must pay.

4 Computationally-Limited Agents

In this section we introduce our model of bounded rationality, in the form of computationally-limited agents. We describe a computationally-limited agent and then explain how we incorporate these agents into a game-theoretic framework.

4.1 A Model of a Computationally-Limited Agent

We define a computationally-limited agent to be any agent who does not know its valuations for items (i.e. its preferences) *a priori* but instead must use its computational resources in order to determine them. We define a computationally-limited agent by its computing resources and the tools that it has to effectively use them.

We assume that a computationally-limited agent has some set of computing resources T_i. While these resources may take on many forms, for the sake of expository ease we will use *computing time* as the canonical example of a resource. An agent is able to apply its computing resources on any problem j in which it is interested in. If there are m possible problems that an agent may compute on, then we let $(t_1, \ldots, t_m) \in T_i^m$ denote that agent i has allocated t_j resources to problem j.

Agents do not have infinite computing resources. We model this limitation by a *cost function*, $\text{cost}_i : T_i^m \mapsto \{x | x \in R \text{ and } x \geq 0\}$. The only restrictions on the cost function of an agent is that it is non-decreasing and additive. That is, given vectors $t = (t_1, \ldots, t_m)$ and $t' = (t'_1, \ldots, t'_m)$, then $\text{cost}_i(t + t') = \text{cost}_i(t) + \text{cost}_i(t')$ and if $t \leq t'$ then $\text{cost}_i(t) \leq \text{cost}_i(t')$.

We assume that computationally-limited agents are equipped with a set of algorithms $\mathcal{A}_i = \{A_i^j\}$ where A_i^j is the algorithm that agent i can use for problem j.[1] These algorithms are processes which the agent runs in order to determine its valuations or preferences. In particular, we assume that all the algorithms have the *anytime property*; they can be stopped at any point in time and will return a solution, and if given additional computational resources, the algorithm will return a better solution. This allows agents to make an explicit tradeoff between the solution quality and the cost to obtain that solution. Many algorithms have the anytime property. For example, most iterative refinement algorithms are anytime since they always return a solution, and improve it if allowed to run longer. Similarly, many search and information gathering applications can be viewed as anytime algorithms. As more information about an item is obtained, the knowledge about its true valuation improves.

While anytime algorithms are models that allow for the trading off of computing resources (for example, computing time) for solution quality, they do not provide a complete solution for agents as they do not specify how this tradeoff should be made. Instead, anytime algorithms are paired with a meta-level deliberation-control procedure the aids in determining how long to run an algorithm, and when to stop computing and act with the solution obtained. There are two components to the procedure; the *performance profile* which describes how computing affects the output of the algorithm, and a process for using the information in the performance profile to make decisions about

[1] In general, an agent can use the same algorithm on multiple problems, or can have multiple algorithms for the same problem.

how much resources to allocate to a problem. For the rest of the paper we will use the term performance profile to refer to both the descriptive and procedural aspects. We use the notation PP_i^j to refer to the performance profile for anytime algorithm A_i^j and let $\mathcal{PP}_i = \{PP_i^j\}$. We will additionally assume that the agents have fully normative performance profiles which allow them to make online decisions about whether to continue computing on a certain problem, whether to stop computing, or whether to switch to computing on a different problem. Such deliberation control procedures do exist. An example is the performance profile tree [4].

To summarize, a computationally-limited agent i is defined as

$$\langle T_i, \text{cost}_i(\cdot), \mathcal{A}_i, \mathcal{PP}_i \rangle.$$

As mentioned at the start of this section, computationally-limited agents do not know their valuations *a priori*. Instead they must compute or gather information to determine them. We assume that the valuation function of an agent ($v(\cdot, \cdot)$) is determined by the amount of resources allocated ($t = (t_1, \ldots, t_m)$) and the allocation (x). That is, the valuation function of agent i is $v_i(t, x)$. The utility of the agent depends on the agent's valuation function, the payment specified by the auction mechanism, and the cost the agent has incurred via computing. That is

$$u_i(t, (x, p)) = v_i(t, x) - p_i - \text{cost}_i(t).$$

For example, in a single item auction, if an agent i has allocated computing resources $(t_1, \ldots, t_m)$ where t_i is the amount of resources allocated to its own valuation problem, then

$$u_i(t, (x, p)) = \begin{cases} v_i(t_i) - p_i - \text{cost}_i(t) & \text{if } i \text{ is allocated the item and has to pay price } p_i \\ -\text{cost}_i(t) & \text{if } i \text{ is not allocated the item} \end{cases}$$

4.2 Strategic Behavior of Deliberative Agents

Effectively using available computing resources in single-agent settings is difficult enough. Having to interact with other agents complicates the problem further. Agents must take into account the other agents' actions in determining both how to act in the auction and also how to compute.

Let C_i be the set of computing actions for agent i. A computing action $c_j^i \in C_i$ is the act of agent i allocating one step of computation on problem j, where a step is defined exogenously. The action of *not* taking a computing step is also included in this set and is denoted by $\emptyset_C$. The vector $(t_1, \ldots, t_m) \in T_i^m$, introduced in the previous subsection, corresponds to agent i taking t_1 steps on problem 1, etc. As agents compute they change their knowledge about how future computing will change the solutions of the problems. This information comes from their set of performance profiles, given their current *state of deliberation*. We define a state of deliberation at $t = (t_1, \ldots, t_m)$ to be

$$\phi_i(t) = \langle n_1(t_1), \ldots, n_m(t_m) \rangle$$

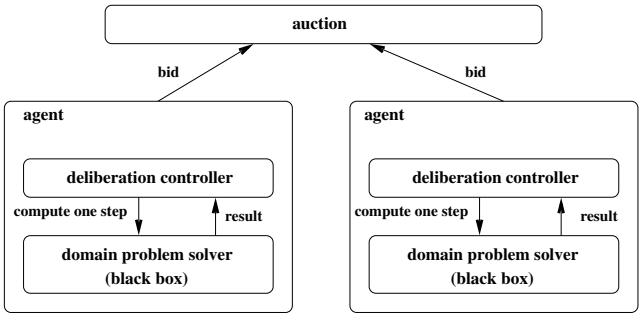

Fig. 2. *An auction with two computationally-limited agents. In order to submit a reasonable bid, each agent needs to first (approximately) compute its valuations for the item that is up for auction.*

where $n_j(t_j)$ stores the current valuation for problem j after t_j computing steps were taken on the problem, as well as the path followed to reach the valuation and any other features deemed to be of importance for the decision making of the agent.[2]

We define the set A_i to be the set of non-deliberative actions that an agent i can take. The set is defined by the auction. For example, in a sealed-bid auction, the set A_i is simply the set of bids that the agent may submit to the auctioneer, while in an ascending auction the set A_i is the set of messages that an agent can send to the mechanism whenever the price increases (i.e. $A_i = \{\text{in}, \text{out}\}$). We denote the action of *not* taking a non-computing action as $\emptyset_A \in A_i$.

A strategy for a computationally-limited agent is a plan which specifies which actions to execute (computing and other) at every point in the game. A *history* at time point r , $H(r) \in \mathcal{H}(r)$, when it is the turn of agent i to take an action, is defined to be a tuple consisting of the current state of deliberation of the agent, all non-computing actions the agent has taken, as well as all actions it has observed the other agents take. It is now possible to define a strategy.

Definition 6 (Strategy). *A strategy for a computationally limited agent i is*

$$S_i = (\sigma_i^r)_{r=0}^{\infty}$$

where

$$\sigma_i^r : \mathcal{H}_i(r) \mapsto C_i \times A_i.$$

To make this definition more clear, we present an example. A strategy $S_i = (\sigma_i^r)_{r=0}^{\infty}$ for a computationally-limited agent participating in a Vickrey auction, where bids are collected at time point R (and it is assumed that all computing must stop also) is defined in the following way:

$$\sigma_i^r(H(r)) = \begin{cases} (c^j, \emptyset_A) & \text{when } r < R \\ (c^j, b_i)\ b_i \in \mathbb{R} & \text{when } r = R \\ (\emptyset_C, \emptyset_A) & \text{when } r > R \end{cases}$$

[2] In the performance profile tree, for example, path information and feature information at automatically stored in the structure.

That is, before the auction the agent can take any computing action it wishes, at the auction bid collection time the agent submits a bid, and then after the agent no longer takes any actions.[3]

In this new, enlarged, strategy space we look for equilibria. We call these equilibria *deliberation equilibria* [4]. We have noted in previous work that a new type of strategic behavior arises when agents are computationally-limited [5]. We have coined this new behavior *strategic deliberation*.

Definition 7 (Strategic Deliberation). *If an agent i uses any of its computing resources to determine another agent's valuation, then agent i is* strategically deliberating.

In earlier work we showed that the first-price sealed-bid auction, the ascending auction, the Vickrey auction and the generalized Vickrey auction all produce equilibria where agents have incentive to strategically deliberate [5].

In the rest of the paper we make several assumptions, some of which have already been described. First, we assume that agents have quasi-linear utilities. Second, we assume that agents have private values. Finally, we assume that the agent definitions are common knowledge. That is, we assume that the performance profiles and cost functions of the agents are common knowledge. We do not assume that agents are able to observe which computing actions other agents are taking during a game.

5 Designing auctions for Deliberative Agents

In this section we study the problem of designing auctions specifically for computationally-limited agents. We first argue that directly porting mechanism design concepts for rational agents is not always desirable as they overlook the computational limitations which define the agents. We then propose a set of properties which we believe auctions for computationally-limited agents should exhibit. We show that most "interesting" auctions fail to have all these properties.

5.1 A Revelation Principle

In settings where the bidding agents are fully rational, the revelation principle states that given any mechanism (auction) that implements some social choice function, it is possible to construct a second incentive-compatible direct-revelation mechanism (auction) which implements the same social choice function. That is, it is possible to construct useful auctions where the dominant strategies of the agents is to truthfully reveal their valuations (types, θ_i) to the auctioneer.

If agents are computationally-limited, then they do not know their valuations, without first exerting some computational effort. Agents are provided with tools (anytime algorithms and performance profiles) which they can use to determine their valuations. We can define the *type* of agent i to be the set of all computing tools that the agent has, along with the current problem instance being computed on. That is,

$$\theta_i = [\langle T_i, \mathrm{cost}_i(\cdot), \mathcal{A}_i, \mathcal{PP}_i \rangle, \mathrm{insts}]$$

[3] In some situations an agent may be permitted to compute after the auction has closes.

where insts is the set of problem instances.[4]

Using this definition of a *type* it is possible to formulate a revelation principle for computationally-limited agents.

Theorem 2. *Suppose there exists a mechanism $M = (S_1, \ldots, S_I, g())$ that implements the social choice function $f(\cdot)$ in dominant strategies for deliberative agents. Then $f(\cdot)$ is truthfully implementable in dominant strategies for computationally-limited agents*

Proof. (**Sketch**.) The proof follows a similar argument to the proof of the revelation principle for rational agents. Given any mechanism, where in equilibrium an agent i plays strategy S_i, it is possible to construct another mechanism, where, when agent i announces type θ_i strategy s_i is executed. Agent i has incentive to truthfully announce its type in equilibrium. □

While this theorem at first appears to suggest that the tools of mechanism-design for rational agents can be directly used in settings where agents are computationally-limited, at second glance the reader will notice that some rather disturbing assumptions have been made. First, the proof relies on agents being able to submit *all* their computing tools and information about the problems they wish to compute on. In practice this is highly infeasible. Second, it assumes that the mechanism center is capable of determining all relevant valuation information from the information given to it. Again, it is unrealistic to assume that the mechanism center will have enough computing resources of its own to actually do this.

5.2 Properties of the Auctions

We believe that auctions for computationally-limited agents should have good deliberative properties in addition to good economic properties. In this section we propose a set of properties which we believe are desirable for computationally-limited agents.

In the previous section we showed that an obvious formulation of a revelation principle simply moves the computing problems of the agents to the auctioneer. We argue that this is not a desirable property since agents should be responsible for solving their own valuation problems. Second, we believe that it is unreasonable to assume that agents are capable of providing detailed enough information to the auctioneer so that the auctioneer could properly solve the problems of the agents. Finally, in many settings the auctioneer is unlikely to have adequate computing resources of its own to solve the computational problems of the bidding agents as well as determine the optimal outcome once all the bidders' computing problems have been solved. Therefore, we propose that auctions should be *non-deliberative*.

Property 1 (Non-Deliberative). An auction is *non-deliberative* if the auctioneer does not solve the agents' individual deliberation problems.

The second property we wish to have is for agents to have no incentive to strategically deliberate (Definition 7). Recall that most common auctions are susceptible to

[4] For example, a problem instance in a traveling-salesman domain is the set of delivery jobs in a traveling-salesman application.

strategic deliberation. We believe that strategic deliberation is an undesirable property for several reasons. First, the use of agents' limited computing resources on problems which do not directly lead to improved valuations seems like a waste of resources which could have been directed towards improving their own valuations, and possibly the social welfare of the entire market. Secondly, effectively using computing resources to determine valuations is a challenging problem for an agent even when it is done in isolation. Having agents being concerned about computing on the valuation problems of competitors seems to place a too high strategic load on the bidding agents. Thus, we believe auctions for computationally-limited agents should be *deliberation-proof*.

Property 2 (Deliberation-Proof). An auction is *deliberation-proof* if, in equilibrium, no agent has incentive to strategically deliberate.

Finally, we believe that auctions should be designed in such a way so that the bidding agents have no incentive to further strategize. That is, agents should have incentive to not misrepresent what problems they have computed on, and what solutions they have obtained.

Property 3 (Non-Deceiving). Assume that v_i is the true (expected) computed value of agent i. A auction is non-deceiving if the agent never has incentive to send a report to the auctioneer such that if any other agent had observed the report, their belief that agent i's value is v_i would be 0.

A non-deceiving auction does not require that an agent directly reveal its computed valuations. It just ensures that an agent does not lead all participants to believe that a valuation it has obtained is not possible. For example, assume that an agent has secretly computed a value v. A mechanism would be deceiving if the agent had incentive to report that its value was strictly greater than v. The mechanism would not be deceiving if the agent had incentive to report that its value was greater than some w, $w < v$.

5.3 Results

In this section we present our results concerning auctions for computationally-limited agents. We will insist that the auctions must be non-deliberative, and so will focus our attention on *value-based* auctions, where agents do not report information about the algorithms or performance profiles being used.

Definition 8 (Value-based Auction). *A value-based auction, $M = (S_1, \ldots, S_n, x(), t_1(), \ldots, t_n())$, is a mechanism where the strategies of each agent are restricted so that they are functions only of (partially) determined valuations. Agents do not reveal other information about performance profiles, algorithms, cost functions or problem instances.*

Value-based auctions are non-deliberative. The auctioneer is not given any of the tools required for it to actively compute on agents' problems. Examples of value-based auctions include sealed-bid auctions where agents submit numerical bids, and ascending and posted-price auctions where agents answer yes or no to the query of whether they would be willing to buy an item at a specified price.

It is trivially easy to design value-based auctions which are deliberation-proof and non-deceiving.

Note 1. There exist value-based auctions which are both deliberation-proof and non-deceiving.

Unfortunately, the obvious candidates which satisfy both of these properties are undesirable for other reasons. For example, any auction which randomly determines a winner while ignoring the bids, is both deliberation-proof and non-deceiving. Similarly, any dictatorial auction is also deliberation-proof and non-deceiving.

We place an additional restriction on the auctions for computationally-limited agents. We insist that they must be *sensitive*.

Definition 9 (Sensitive). *An auction is* sensitive *to agents' strategies if the outcome, $o() = (x(), p())$ of the auction depends on the agents' strategies. That is, there exists some agent i and strategies s'_i, s''_i, $s'_i \neq s''_i$ such that for strategy profiles $s' = (s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_I)$ and $s'' = (s_1 \ldots, s_{i-1}, s''_i, s_{i+1}, \ldots, s_I)$,*

$$x(s') \neq x(s'') \text{ and } p(s') \neq p(s'')$$

In the rest of this paper we focus our attention to sensitive, value-based mechanisms. We start by studying sealed-bid auctions to see how computationally-limited agents behave in equilibrium. The first result states that it is impossible to design a sealed-bid auction which is also deliberation-proof.

Theorem 3. *There exists no value-based sensitive direct auction that is deliberation-proof across all instances (where an instance is defined by the agents and the current valuation problems).*

Proof. Due to space constraints we only provide an intuition of the proof. Given any allocation and payment rules of a sealed-bid auction, it is possible to construct performance profiles and cost functions such that one agent is best off incurring a small cost by computing on a competitor's valuation problem, and thus gathering information as to the likelihood of it being in the final allocation, before deciding whether to compute at a cost on its own valuation problems. □

To get rid of strategic deliberation, computationally-limited agents must be provided with enough information by the auction so that they can determine whether to devote computing resources to their own problems or not. Many multi-stage auctions reveal information. This information can be used by the agents to help determine which are the best computing and non-computing actions to take. For example, in a single item ascending auction, as the price rises the auction may reveal the number of agents remaining in the auction at a given price. The agents can use this information to deduce useful valuation information about their competitors.

However, value-based multi-stage auctions still do not result in auctions with all our proposed properties. In particular, for any multi-stage auction, there exist instances, defined by the agents' performance profiles and cost functions, where strategic deliberation occurs in equilibrium, or where agents will try to deceive their competitors by taking actions which lead their competitors to believe that they have better valuations than they really do.

Theorem 4. *There exists no sensitive value-based auction that is deliberation-proof and non-deceiving across all problem instances. (An instance is defined by agents' performance profiles, cost functions).*

Proof. Due to space limitations we provide only an overview of the proof. The proof is constructive. Starting with a direct revelation mechanism (auction) which implements some social choice function $f(\cdot)$, it is possible to find performance profiles and cost functions such that strategic deliberation occurs in equilibrium. Now take any multi-stage mechanism which implements the social choice function $f(\cdot)$. It is still advantageous for at least one agent to learn information about the values another agent has. If this information is not revealed at the right time through the auction, then strategic deliberation will still occur. If, by the rules of the auction, the actions of an agent reveal information about its possible values, then due to the structure of the performance profiles and cost functions, it is possible to show that one agent will misreport its value information in order to force the second agent (ie. the agent who would have strategic deliberated in a direct mechanism) to believe that it can not win the auction. That is, deceiving occurs. □

To summarize, we have proposed that an auction for computationally-limited agents should be non-deliberative, deliberation-proof, and non-deceiving. While these properties appear to be intuitive and reasonable, we have shown that it is not possible to design interesting auctions for deliberative agents which exhibit all three properties.

6 Related Research

Both the the game theory and the computer science research communities are interested in mechanism design issues where computation and information gathering are issues. In this section we review the literature which is most closely related to the work presented in this paper.

From the computer science research community there has been work on both bounded-rational bidding agents and mechanisms. Sandholm noted that under a model of costly computing, the dominant strategy property of Vickrey auctions fails to hold, even with simple agent models [12], while Parkes has looked at using auction design to simplify the meta-deliberation problems of the agents [9]. This earlier work, however, focused only on settings where agents were concerned only about computing on their own value - ignoring the possibility of using computational resources to gather information on competing bidders. There has also been recent work on computationally limited mechanisms. In particular, research has focused on the generalized Vickrey auction and investigates ways of introducing approximate algorithms to compute outcomes without loosing the incentive compatibility property [7, 3, 6]. These methods still require that the bidding agents compute and submit their valuations.

In the economics and game theory literature there has been some recent work on information acquisition and mechanism design. This work has mainly focused on studying the incentives to acquire information at different times in different auction mechanisms and usually assumes that an agent can gather information only on its value problem [10, 1, 2]. Rasmusen's work is the most similar to ours [11]. It assumes that agents

do not know their valuations but must invest to learn them and are also able to invest in competitors value problems, however his focus is on understanding behavior such as sniping that is commonly seen in different online auctions like eBay, and he does not discuss the possibility of designing auctions for information gathering settings.

7 Conclusions and Future Research

In many settings, agents participating in mechanisms do not know their preferences (valuations) *a priori*. Instead, they must actively determine them through deliberation (e.g., information processing or information gathering). Agents are faced not only with the problem of deciding how to reveal their preferences to the mechanism but also how to deliberate in order to determine their preferences. For such settings, we have introduced the *deliberation equilibrium* as the game-theoretic solution concept where the agents' deliberation actions are modeled as part of their strategies.

In this paper, we laid out mechanism design principles for such deliberative agents. We first showed that the revelation principle applies to such settings in a trivial sense by having the mechanism carry out all the deliberation for the agents. This is impractical, and we propose that mechanisms should be *non-deliberative*: the mechanism should not be solving the deliberation problems for the agents. Second, mechanisms should be *deliberation-proof*: agents should not deliberate on others' valuations in equilibrium. Third, the mechanism should be *non-deceiving*: agents do not strategically misrepresent. Finally, the mechanism should be *sensitive*: the agents' actions should affect the outcome. We showed that no direct-revelation mechanism satisfies these four intuitively desirable weak properties. This is the first impossibility result in mechanism design for deliberative agents. Moving beyond direct-revelation mechanisms, we showed that no *value-based* mechanism (that is, mechanism where the agents are only asked to report valuations - either partially or fully determined ones) satisfies these four properties.

This result is negative. It states that either we must have mechanisms which do the deliberating for the agents, or complex strategic (and costly) counterspeculation can occur in equilibrium. However, there is some hope. It may be possible to weaken one of the properties slightly, while still achieving the others. For example, it may be possible to design multi-stage mechanisms that are not value based; the mechanism could help each agent decide when to hold off on deliberating during the auction (and when to deliberate on one's own valuation on different bundles of items in a combinatorial auction). In another direction, by relaxing strategic deliberation and compensating agents appropriately, it may be possible to design mechanisms where agents who can deliberate cheaply and efficiently deliberate for all agents. These are areas which we plan to pursue in future work.

References

1. Olivier Compte and Philippe Jehiel. On the virtues of the ascending price auction: New insights in the private value setting. working paper, December 2000.
2. Olivier Compte and Philippe Jehiel. Auctions and information acquisition: Sealed-bid or dynamic formats?, 2001. Working Paper.

3. Noa Kfir-Dahav, Dov Monderer, and Moshe Tennenholtz. Mechanism design for resource bounded agents. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS)*, pages 309–315, Boston, MA, 2000.
4. Kate Larson and Tuomas Sandholm. Bargaining with limited computation: Deliberation equilibrium. *Artificial Intelligence*, 132(2):183–217, 2001. Short early version appeared in the Proceedings of the National Conference on Artificial Intelligence (AAAI), pp. 48–55, Austin, TX, 2000.
5. Kate Larson and Tuomas Sandholm. Costly valuation computation in auctions. In *Theoretical Aspects of Rationality and Knowledge (TARK VIII)*, pages 169–182, Siena, Italy, July 2001.
6. Daniel Lehmann, Lidian Ita O'Callaghan, and Yoav Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002. Early version appeared in ACMEC-99.
7. Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001. Early version in STOC-99.
8. Christos Papadimitriou. Algorithms, games and the Internet. In *STOC*, pages 749–753, 2001.
9. David C. Parkes. Optimal auction design for agents with hard valuation problems. In *Agent-Mediated Electronic Commerce Workshop at the International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 1999.
10. Nicola Perisco. Information acquisition in auctions. *Econometrica*, 68(1):135–148, January 2000.
11. Eric Rasmusen. Strategic implications of uncertainty over one's own private value in auctions. working paper, January 2003.
12. Tuomas Sandholm. Issues in computational Vickrey auctions. *International Journal of Electronic Commerce*, 4(3):107–129, 2000. Special Issue on Applying Intelligent Agents for Electronic Commerce. A short, early version appeared at the Second International Conference on Multi–Agent Systems (ICMAS), pages 299–306, 1996.