Costly Valuation Computation in Auctions

Kate Larson and Tuomas Sandholm *
Computer Science Department
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213 USA
{klarson, sandholm}@cs.cmu.edu

Abstract

We investigate deliberation and bidding strategies of agents with unlimited but costly computation who are participating in auctions. The agents do not *a priori* know their valuations for the items begin auctioned. Instead they devote computational resources to compute their valuations. We present a normative model of bounded rationality where deliberation actions of agents are incorporated into strategies and equilibria are analyzed for standard auction protocols. We show that even in settings such as English auctions where information about other agents' valuations is revealed for free by the bidding process, agents may still compute on opponents' valuation problems, incurring a cost, in order to determine how to bid. We compare the costly computation model of bounded rationality with a different model where computation is free but limited. For some auction mechanisms the equilibrium strategies are substantially different. It can be concluded that the model of bounded rationality impacts the agents' equilibrium strategies and must be considered when designing mechanisms for computationally limited agents.

1 Introduction

Systems, especially on the Internet, are increasingly being used by multiple parties—or software agents that represent them—with their own preferences. This invalidates the traditional assumption that a central designer controls the behavior of all system components. The system designer can only control the *mechanism*, while each agent chooses its own *strategy*. The economic efficiency that a system yields depends on the agents' strategies. To develop a system that leads to desirable outcomes, the designer has to make sure that each agent is motivated in the desired way. This can be achieved by analyzing the game using the *Nash equilibrium* solution concept from game theory (or its refinements). However, the equilibrium for rational agents does not generally remain an equilibrium for computationally limited agents. This leaves a potentially hazardous gap in game theory, as well as automated negotiation, since computationally limited agents are not motivated to behave in the desired way.

Early on, it was recognized that humans have bounded rationality, e.g., due to cognitive limitations, so they do not act rationally as economic theory would predict [23]. Since then, considerable work has focused on developing *normative* models that prescribe how a computationally limited agent *should* behave [1, 2, 18]. However, this has been a problem filled with many technical difficulties and so simplifying assumptions have been used.

*This material is based upon work supported by the National Science Foundation under CAREER Award IRI-9703122, Grant IIS-9800994, and ITR IIS-0081246.

While such simplifications are acceptable in single-agent settings as long as the agent performs reasonably well, any deviation from full normativity can be catastrophic in multiagent settings. If the designer cannot guarantee that the strategy (including deliberation actions) is the best strategy that an agent can use, there is a risk that an agent is motivated to use some other strategy. Even if that strategy happens to be "close" to the desired one, the social outcome may be far from desirable.

Game theorists have also realized the significance of computational limitations, but have focussed on how complex it is to compute the rational strategies [9], memory limitations in keeping track of history [17], or limited uniform–depth lookahead capabilities in repeated games [7]. Instead, we are interested in settings where agents' limited rationality stems from the complexity of the agents' optimization problems, a setting which is ubiquitous in practice [22].

Game theory is a useful tool to help in the design of bidding agents for auctions. There is a vast literature describing *rational* agents' optimal bidding strategies in different types of auctions [13]. However, economic models for *bounded–rational* agents have often been descriptive rather than prescriptive [23, 17]. In order to provide a prescriptive model, Larson and Sandholm proposed incorporating deliberation actions into agents' strategies in order to analyze, game theoretically, bounded–rational agents in a 2–agent bargaining game [11].

In auctions there has been work on both bounded–rational bidding agents and mechanisms. For bounded–rational bidding agents, Sandholm noted that under a model of costly computation, the dominant strategy property of Vickrey auctions fails to hold [21]. Instead, an agent's best deliberation action can depend on the other agents. In recent work, auction settings where agents have hard valuation problems have been studied [16]. Auction design is presented as a way to simplify the meta–deliberation problems of the agents', with the goal of providing incentives for the "right" agents to deliberate for the "right" amount of time. A costly computation model, with simple deliberation control where agents compute to refine their valuations, is used. However, situations where agents may compute on each others' problems in order to refine their bids are not considered, nor are combinatorial auctions, where agents have to select which of their own valuation problems and which of their opponents' valuation problems to compute on, studied.

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 [15, 8, 12]. These methods still require that the bidding agents compute and submit their valuations.

In this paper we investigate deliberation and bidding strategies of agents with unlimited but costly computation who participate in auctions. They do not *a priori* know their valuations for the items being auctioned. Instead, they must devote computational resources in order to compute their valuations. Agents are also allowed to compute on other agents' valuation problems so as to gain information about potential bidding strategies. We present a fully normative model of deliberation control and define agents' strategies in this setting, as well as the concepts of strong and weak strategic computation. We analyze different auction protocols, presenting results on whether strategic computation occurs in equilibrium. We conclude with a comparison to a setting where computation is

limited but free, and provide a discussion how the model of bounded rationality affects the equilibrium strategies of agents.

2 Auctions

Auctions provide efficient and distributed ways of allocating goods and tasks among agents. In this paper we investigate inherently private value auctions with risk–neutral agents. There are many different auction mechanisms, but in this paper we focus on the single item English, Dutch, Vickrey, and first–price sealed–bid auctions. We also discuss combinatorial auctions where bids can be submitted on bundles of items. In particular, we focus on a sealed–bid combinatorial auction, (Generalized Vickrey auction (GVA)), where with an expressive enough bidding language [20], truthful bidding is a dominant strategy using the Groves–Clarke mechanism [4, 3].

3 The Roles of Computation

To participate in an auction, agents need to determine valuations for the items being auctioned. The question becomes: how are these valuations derived? In this paper we focus on situations where agents do not simply know their own valuations. Rather, they have to allocate computational resources to compute the valuations. We assume all agents have *anytime algorithms* so that a result can be supplied at any time, and solution quality increases with time. This enables each agent to make a tradeoff between time and solution quality. Anytime algorithms can be broadly classified into two groups. The first group includes iterative improvement algorithms such as iterative refinement. These algorithms always have a solution at hand and make small changes to the solution to try and improve it. Iterative improvement algorithms have been very successful in finding nearly optimal solutions to hard problems with very small amounts of time [14]. The second group of anytime algorithms contains complete algorithms such as many tree search algorithms. Branch and bound, for example, improves the bounds on the solution as more time is allocated to the algorithm [19].

We present two roles where agents use computation.

Role 1: Computation improves valuations In our first model, computation increases the agents' valuations. As the agent computes further, the agent finds better ways of using the items. Therefore, the agent may be willing to place a "better" bid.

An example is a procurement auction where agents are companies bidding on a task which consists of delivering a parcel from one location to another. The agent who submits the lowest bid wins and its utility is the bid amount, b_i, that it gets paid minus its cost, c_i, of performing the delivery. As the agent computes on the problem of how to deliver the parcel, it can obtain better (less costly) vehicle routing solutions. The agent might then want to modify its bid, decreasing it so as to increase the likelihood of winning the auction.

Role 2: Computation refines the agent's belief Alternatively, computation can refine an agent's valuation. The agent can maintain a distribution (or bounds) on the valuation. Additional computation can then refine the

agent's beliefs, causing the distribution of the value to shift up or down.

4 Costly Computation and Normative Control of Deliberation

If agents know their valuations *a priori*, or are able to compute their valuations with ease, they would execute the equilibrium bidding strategies for rational agents. However, agents often have limitations on their deliberation resources which may take the form of costly or limited (deadlines and finite processing speed) computation.

The agents must decide how to allocate their limited deliberation resources. In single–item settings they must decide whether to compute only on their own valuation problem in order to obtain the best valuation possible, or whether to use some of their computational resources on other agents' valuation problems. In combinatorial auctions the agents' decisions become more complex. Each agent must decide how to split its computation across different bundles it is interested in as well as bundles the other agents might place bids on.

In this paper we consider situations where agents have the possibility of unlimited computation, but each computation step incurs a cost. Each agent has a cost function $c_i : T^n \rightarrow \mathbb{R}$, where T is time and n is the number of problems an agent may compute on. In single item auctions, n equals the number of bidders and in combinatorial auctions, n is the number of bidders times 2^m, where m is the number of goods being auctioned.[1] At time $\bar{t} = (t_1, t_2, \ldots, t_n)$, the utility of agent i from computing t_j time steps on good g_j is $u_i^{g_j} = v_{g_j}(t_j) - c_i(\bar{t})$ where $v_{g_j}(t_j)$ is the valuation computed for the good. Note that an agent may have different computational costs for different problems.

Each agent has an *anytime algorithm* that has a feasible solution available whenever it is terminated, and improves the solution as more computation time is allocated to the problem. Let $v_\alpha^g(t)$ be the value agent α has obtained after devoting t computation steps to the valuation problem for good g. The agents have statistical *performance profiles* that describe how computation changes the valuations. Each agent uses this information to decide how to allocate its computation at every step in the process, based on the results of its computation so far.

There has been much work on performance profile based deliberation control [24, 5, 2, 6]. To represent the performance profiles we use a tree structure [11]. The advantage of this approach is that it allows optimal conditioning on results of execution so far, and allows conditioning on the actual problem instance.

We specify two different types of performance profiles, *stochastic* and *deterministic*. In a *stochastic performance profile* there is uncertainty as to what results future computation will bring. At least one node in the tree has multiple children. The uncertainty can come from variation in performance on different problem instances or from the use of a randomized algorithm. In a *deterministic performance profile* the algorithm's performance can be projected with certainty (i.e. the tree is a branch). Before doing any computation, an agent can determine what the solution will be after any number of computation steps devoted to the problem.

We index the problem by i and g where i is an agent and g is an item (or bundle of items) in the auction. For each g and i there is a performance profile tree, $\mathcal{T}_i^g$. This represents the fact that the tree may be conditioned on

[1] This takes into account the fact that agents' valuations for the empty bundle need not be 0.

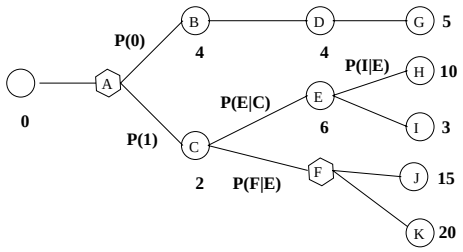

Figure 1: *An agent's stochastic performance profile tree for a valuation problem. The hexagonal nodes are random nodes and the round nodes are value nodes. At random node A, the probability that the random number will be 0 is $P(0)$, and the probability that the random number will be 1 is $P(1)$. The edges from any value node have a weight of one time step, while the edges from any random node have a weight of zero time steps. This means it takes 3 time steps to reach node G, and 2 time steps to reach node J.*

features of the problem instance. Figure 1 exemplifies one such tree.

The general tree form represents uncertainty that comes from both randomized algorithms and variation of performance on different problem instances. There are two different types of nodes in the performance profile tree, random number nodes and value nodes. A random node occurs whenever a random number is needed to chart the path of the algorithm run. For example, an anytime algorithm for the Travelling Salesman Problem (TSP), randomly chooses cities and swaps them in the route. In this situation, a random number would be a city label. The edges from a random node represent the possible random numbers (bits), and are labeled with the probability of a certain random number occurring. Each edge emanating from a random node has a weight of zero. Value nodes store the value of the solution that the algorithm has computed at a certain point in time. The children of a value node may be either more value nodes or random nodes. Each edge emanating from a value node has a weight of one time step. The edges are labeled with the probability of reaching the child, given that the parent was reached. This allows one to compute the probability of reaching any particular future node in the tree given any node, by multiplying the probabilities on the path between nodes. If the only uncertainty in the algorithm's run is from the variation of performance on different problem instances, then there are no random nodes in the performance profile tree.

We denote by $\mathrm{time}(n)$ the sum of the weights of the edges that were followed to reach the node n. In other words, $\mathrm{time}(n)$ is the number of computational steps used to reach node n. We denote by $V(n)$ the value of a value node n.

In practice it is unlikely that an agent knows the valuation for every time allocation without actually doing the computation. Rather, there is uncertainty about how the valuation improves over time. A performance profile tree allows one to capture this uncertainty. The tree can be used to determine $P(v_i^g|t)$ denoting the probability that running the algorithm for t time steps produces a solution of value v_i^g.

Unlike earlier performance profile models for controlling anytime algorithms (e.g., [24, 2, 6, 5]), the performance profile tree supports conditioning on the path of valuation quality so far. The performance profile tree that applies given a path of computation is the subtree rooted at the current node n. This subtree is denoted by $\mathcal{T}_i^g(n)$.

If an agent is at a node n with value v, then when estimating how much additional deliberation would increase the valuation, the agent need only consider paths that emanate from node n. The probability, $P_n(n')$, of reaching a particular future node n' in $\mathcal{T}_i^g(n)$ is simply the product of the probabilities on the path from n to n'. The expected valuation after allocating t more time steps to the problem, if the current node is n, is

$$\sum P_n(n') \cdot V(n')$$

where the sum is over the set $\{n' | n'$ is a node in $\mathcal{T}_i^g(n)$ which is reachable in t time steps$\}$.

In our equilibrium analysis, we assume that all performance profile trees are common knowledge and that agents have access to the same algorithms. Agent i can compute on agent j's valuation for item g, using the performance profile $\mathcal{T}_j^g$ to guide its computation. However, this computation would not change agent i's valuation for good g. The most practical model is the special case where the agents have the same performance profiles, i.e., $\mathcal{T}_i^g = \mathcal{T}_j^g$ for all agents i and j. This models a real world setting where the agents have been solving similar problems in the past, and have been improving their algorithms for those problems resulting in comparable algorithms among the agents.

If the only uncertainty comes from the variation in the performance of the algorithm on different problem instances, then agents can compute on others problems, and are able to obtain identical results. This information can be used when it comes to the bid formation process. If an agent can determine that another agent has a high valuation for a good, then it might decide that the likelihood of obtaining the good in the auction is small, and thus can concentrate its resources on obtaining valuations for other items. However, if agents are running randomized algorithms, then, since the path is dependent on random numbers, one agent's results from computing on a certain valuation problem may be different from the valuation obtained by another agent computing on the same problem. Therefore, we allow one agent to *emulate* the possible algorithm runs another agent may be following. While emulating an algorithm run, whenever an agent reaches a point where a random number is used, the agent can select which random number to run the algorithm with. At a later point in time, the agent may revisit the same random node, and select a different random number with which to run the algorithm. While the emulation does not remove all uncertainty as to what valuations other agents have computed (as the actual random numbers are not known), it does allow an agent to remove some uncertainty as it provides information about the possible paths the algorithm might have taken.

Agents store the results of their deliberation actions at each time step in a state of deliberation. The state of deliberation also stores the cost incurred from computing.

Definition 1 *The* **state of deliberation** *of agent i at time t when there are no randomized algorithms is*

$$\theta_i(t) = (\langle n(j,g) \rangle_{j \in A}^{g \in G}, c_i(\bar{t}))$$

where A is the set of agents participating in the auction, G is the set of bundles being auctioned, $\sum_{g,j} \mathrm{time}(n(j,g)) = t$, and $\bar{t} = (\mathrm{time}(n(j,g)))_{g \in G}^{j \in A}$.

If there are randomized algorithms then the state of deliberation of agent i at time t is

$$\theta_i(t) = (\langle n(j,g,r) \rangle, c_i(\bar{t}))$$

where $g \in G$, $j \in A$, r is a (set of) random number used for the problem specified by g and j, $\sum_{g,j} \sum_r \text{time}(n(j,g,r)) = t$, and $\bar{t} = ((\sum_r \text{time}(n(j,g,r)))_{j \in A}^{g \in G})$.

5 Bidding Strategies and Strategic Computation

A strategy for an agent is composed of two interrelated components – the deliberation component and the bargaining component. What an agent bids depends on the solutions it has obtained for its valuation problems, and what an agent computes on depends partially on what bids it has observed.

Let A be the set of agents participating in the auction and G be the set of bundles. Let $\text{Act}(A, G)$ be the set of deliberation actions where $\text{act}_i^g \in \text{Act}(A, G)$ is the action of (possibly choosing a random number and then) taking one computation step on agent i's problem for bundle g. We do not require that agents compute every step. Instead, they may decide to pass on computing. Agents do not incur any cost when they decide not to compute on a problem. We denote by $\emptyset$ the act of taking no deliberation step. While bidding, agents may either send a message stating how much their bid is, or that they no longer wish to participate in the auction. We require agents to send "withdraw" messages in sequential auctions in order to differentiate between not increasing a bid, and leaving the auction.

The auction mechanism determines the agents' strategies.

Definition 2 *A strategy for agent i in a sequential auction is*

$$S_i = ((\sigma_i^d(t), \sigma_i^b(t)))_{t=0}^{\infty}$$

with deliberation component

$$\sigma_i^d(t+1) : \theta_i(t) \times B_i(t) \to \text{Act}(A, G)$$

where $B_i(t)$ is the list of bids observed by agent i up to time t. The bidding component is

$$\sigma_i^b(t+1) : \theta_i(t+1) \times B_i(t) \to \mathbb{R}^{2^M - 1} \cup \{\text{withdraw}\}.$$

In a single–shot setting the strategy is slightly different. The auction ends at some time T at which point agents submit a single bid. An agent never has the opportunity to observe the other agents' bids.

Definition 3 *A strategy for agent i is a single–shot auction which closes at time T is*

$$S_i = ((\sigma_i^d(t))_{t=0}^{\infty}, \sigma_i^b)$$

where the deliberation component is

$$\sigma_i^d(t+1) : \theta_i(t+1) \to \text{Act}(A, G) \quad \textit{and} \quad \sigma_i^b : \theta_i(T) \to \mathbb{R}^{2^M - 1}.$$

If agents' deliberation and bidding strategies are in (Nash, perfect Bayesian, etc.) equilibrium then we say there exists a (Nash, perfect Bayesian, etc.) **deliberation equilibrium**.

Agents use their computational resources in different ways. They can compute on their own problems in order to obtain better valuations. They can also compute on their opponents' problems in an attempt to gather information about the bids that the opponents may be submitting. We make a distinction between these two types of deliberation. The first we call *weak strategic computation*. The second we call *strong strategic computation*.

Definition 4 *If an agent i uses part of its deliberation resources to compute on another agent's valuation problems, then agent i is performing* **strong strategic computation**.

Definition 5 *If an agent i does not actually use its deliberation resources to compute on another agent's valuation problems, but does use information from the opponents performance profile to devise a strategy, then agent i is performing* **weak strategic computation**.

In strong strategic computation the agent uses its own deliberation resources to compute on opponents' problems, thus incurring a higher cost than if it deliberated on its own valuation problems. This increased cost, may cause an agent to compute less on its own valuation problems. In weak strategic computation, an agent does not use its computational resources to compute on an opponent's valuation problem. Instead it forms its deliberation and bidding strategies based on information obtained by examining the opponent's performance profiles. Ideally, neither form of strategic computation is present in an auction. However, strong strategic computation is the least desirable since agents not only counterspeculate on other agents' strategies, but also use their own limited resources in the process, leading to higher cost and less computation time (and therefore worse solutions) on their own actual problems. In economic terms, strong strategic computation generally decreases Pareto efficiency.

6 Results: Does Strategic Computation Occur in Equilibrium?

For some auction mechanisms, the occurrence of strong strategic computation does not come as a surprise. In particular, for first–price, sealed–bid and Dutch auctions where rational agents counterspeculate, it is straightforward to see that if the cost of obtaining information about other bidders is small, agents may be willing to invest their resources in order to determine what the other agents' valuations might be.

The following results formalize this idea. The occurrence of strong strategic computation depends on the performance profiles and the cost functions of the agents.

Theorem 1 *Strong strategic computation can occur in Nash equilibrium in first price sealed bid auctions.*

Due to limited space, proofs for Theorems 1 to 4 are omitted from this extended abstract.

Theorem 2 *Strong strategic computation can occur in Nash equilibrium in Dutch auctions.*

If an agent can pay to remove uncertainty about its valuation, then it may counterspeculate its opponents in the Vickrey auction (i.e., perform weak strategic computation) [21]. Thus, the Vickrey auction ceases to have the dominant strategy property. However, if the performance profiles are stochastic, then depending on the performance

profiles and cost functions, in equilibrium agents may devote some of their deliberation resources on computing other agents' valuation problems.

Theorem 3 *Strong strategic computation can occur in Nash equilibrium in Vickrey auctions if the performance profiles are stochastic. If the performance profiles are deterministic (i.e., branches), then only weak strategic computation will occur.*

The costly computation also affects the equilibrium strategies of agents participating in auctions where there are multiple items and thus multiple valuation problems on which to compute. If the agents have costly computation, the GVA looses its incentive compatibility property.

Theorem 4 *If the performance profiles are stochastic, strong strategic computation can occur in Nash equilibrium in generalized Vickrey auctions. If the performance profiles are deterministic (i.e. branches), no strong strategic computation can occur in Nash equilibrium in generalized Vickrey auctions. However, weak strategic computation can occur.*

The most interesting results occur in English auctions. In an English auction, information is revealed as the auction progresses. As the bids increase, participants are able to update their beliefs about the other agents' valuations and are able to determine whether they wish to continue participating in the auction

In situations with fully rational agents, no form of strategic computation occurs in equilibrium. A fully rational agent will increment its bid whenever it does not have the highest bid, and the highest bid is less than the agent's valuation. Enough information is revealed through the auction mechanism so that participants have no need to counterspeculate or learn their opponents' valuations through any method other than waiting and observing. With a costly computation model, agents' strategies are different. In equilibrium, strong strategic computation can occur.

There are some instances in which no strong strategic computation will occur in equilibrium.

Theorem 5 *If at most one agent's performance profile is stochastic, then in an English auction no strong strategic computation will occur. Agents have dominant strategies.*

Proof: If an agent has a deterministic performance profile, then it has a dominant strategy. It participates in the auction by incrementing its bid whenever it does not have the highest bid, b, and

$$\max_t [v_i(t) - c_i(\bar{t}_i)] - b \leq 0$$

where $\bar{t}_i = (t_1, t_2, \ldots, t_{|A|})$ such that $t_i = t$ and for all j such that $i \neq j$, $t_j = 0$. It does not perform any computation until the auction has closed and it has won the auction.

Assume one agent has a stochastic performance profile. It's strategy depends on how the other participants bid. However, it can gain nothing by computing on its opponents' valuation problems as it already has all information from just examining the performance profiles without computing. □

If more than one agent has a stochastic performance profile then strong strategic computation can occur. It may be better for an agent to compute on an opponent's valuation problem in order to determine whether to remain in

an auction or to withdraw early without fully resolving its own valuations. Subtle aspects of performance profiles distinguish whether strong strategic computation occurs in equilibrium.

Theorem 6 *In an English auction there exist instances where strong strategic computation does and does not occur in Nash equilibrium (even with the same performance profile topologies across those instances).*

Proof: Assume that there are two agents, agent α and agent β who have the performance profiles in Figure 2. Assume that $c_\alpha(\bar{t}) = 0$ and $c_\beta(\bar{t}) = t_\alpha + t_\beta$ for all $\bar{t} = (t_\alpha, t_\beta)$. Agent α has a dominant strategy. It computes one step on its own problem (at a cost of 0) to determine whether its valuation is high (h_α) or low (l_α). It will bid in the auction until the bid is higher than its valuation. It gains nothing by computing on agent β's problem. For each

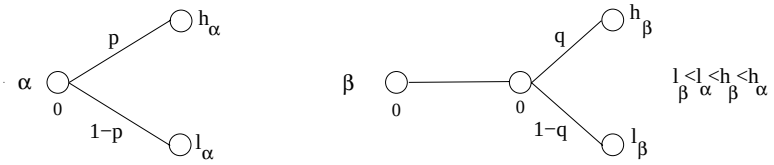

Figure 2: *Performance profiles for agents α and β.*

computation step that agent β takes, it incurs a cost of 1. Therefore agent β must decide whether to compute on a problem and incur the cost, or not. It may be to agent β's advantage to compute on agent α's valuation problem in order to determine whether it wants to continue participating in the auction and devote computational resources to its own valuation problem. What agent β decides to do depends on agent α's bids. Let b_i be agent i's highest bid.

Case 1. If $b_\alpha > l_\alpha$ then agent α has valuation h_α and agent β knows this. Agent β will not be able to out bid agent α. Agent β has no incentive to participate in the auction since it is impossible for it to win the item. Agent β has no incentive to compute further on its own valuation problem nor on agent α's valuation problem since it has already learned that agent α has a high valuation. Therefore agent β withdraws from the auction.

Case 2. If $b_\alpha < l_\beta - 1$ then agent β continues to increment its bid as long as agent α does the same. Agent β will not compute on any valuation problem as it knows that its own valuation for the item is not less than l_β and the cost to compute this valuation is 1. If agent α withdraws while $b_\beta < l_\beta - 1$ then agent β will compute on its own problem.

Case 3. If $l_\beta - 1 \leq b_\alpha \leq l_\alpha$ then agent β's best–response strategy is more complex. When $b_\alpha = l_\beta - 1$, agent β must decide whether to continue bidding without knowing its true valuation, whether to compute on its own problem to determine its valuation, whether to compute on agent α's problem to determine whether agent α has high valuation, or whether to withdraw from the auction. These decisions are captured in Figure 3. If agent β computes on agent α's problem then strong strategic computation occurs. The question becomes, does there exists parameters on the performance profile trees such that agent β will compute on agent α's problem? That is, does there exist p, $q \in [0, 1]$ and h_β, l_β, h_α and l_α such that

$$2p + q(h_\beta - l_\alpha) + pq(l_\alpha - h_\beta) - 3 > q(h_\beta - l_\alpha) + pq(l_\alpha - h_\beta) - 2 \quad (1)$$

$$2p + q(h_\beta - l_\alpha) + pq(l_\alpha - h_\beta) - 3 > p(l_\alpha - l_\beta + 2) + pq(l_\beta - h_\beta) + l_\beta - l_\alpha - 2 \quad (2)$$

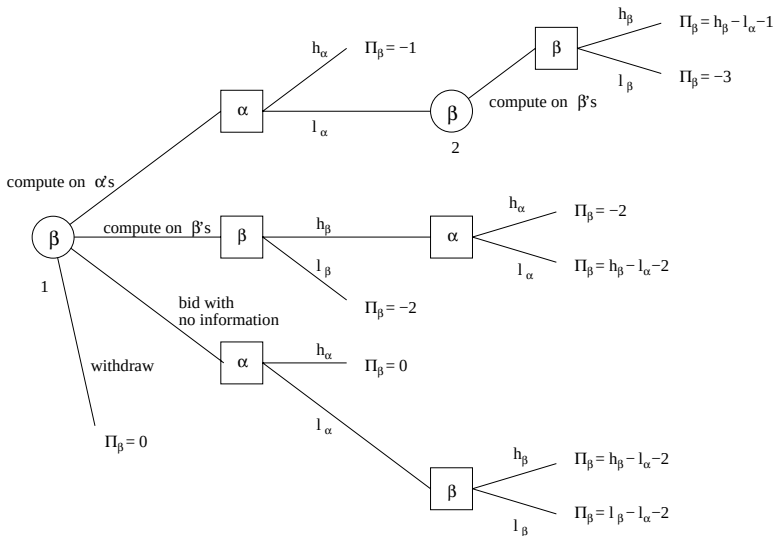

Figure 3: *Agent β's decision tree. The circles represent times when β must take an action. The squares represent times when nature moves and the agents' valuations are revealed. Agent α has valuation h_α with probability p and agent β has valuation h_β with probability q. Only undominated choices are shown. In particular, at node 2, β has more than the one choice shown. It could also withdraw from the auction or bid before determining its true valuation. However, both these actions result in β having strictly lower utility than from what it could obtain if it computed on its own problem. Therefore, they were not included in the decision tree.*

$$2p + q(h_\beta - l_\alpha) + pq(l_\alpha - h_\beta) - 3 > 0. \tag{3}$$

We obtain the following constraints.

$$\frac{1}{2} < p < 1, \quad 0 < q < 1$$

$$h_\alpha > \frac{3 - 2p - 2q + 2pq}{q - pq - q^2 + pq^2}, \qquad \frac{1}{1 - p - q + pq} < l_\alpha < \frac{3 - 2p - h_\alpha q + h_\alpha pq}{pq - q}$$

$$\frac{-3 + 2p - l_\alpha q + l_\alpha pq}{pq - q} < h_\beta < h_\alpha, \qquad 0 \le l_\beta < \frac{-1 + l_\alpha - l_\alpha p - l_\alpha q + l_\alpha pq}{1 - p - q + pq}. \qquad \square$$

We present an example where strategic computation occurs, depending on the probability distributions of the valuations. Consider the performance profile trees of Figure 2. Let $h_\alpha = 30$, $l_\alpha = 12$, $h_\beta = 22$ and $l_\beta = 3$. Figure 4 shows the ranges for the probabilities p and q where agent β's optimal strategy involves computing on agent α's valuation problem.

7 Comparison of Different Models of Computation

Another model of limited computation is one where computing is free but agents have deadlines after which they may no longer compute. The effects of this model on deliberation and bidding strategies has been previously studied, [10]. Interestingly, the equilibrium behavior of the agents are different in the two models. Table 1 summarizes

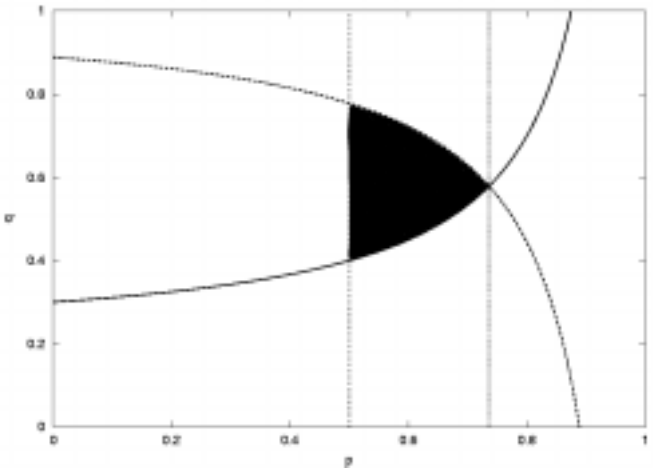

Figure 4: *Graph illustrating when strong strategic computation will occur in equilibrium. The shaded region contains the values for p and q where agent β will compute on agent α's valuation problem. The unshaded region contains the values for p and q where strong strategic computation will not occur in equilibrium.*

the differences and compares the computationally limited agents' strategies with those of fully rational agents, depending on whether counterspeculatation would occur in equilibrium or not.

	Auction mechanism	Counterspeculation by rational agents?	Strategic computation?	
			Limited computation	**Costly computation**
Single item	Vickrey	no	no	**yes**
	English	no	no	**yes**
	First Price	yes	yes	**yes**
	Dutch	yes	yes	**yes**
Multiple items	GVA (deterministic performance profiles)	no	yes (weak only)	**yes (weak only)**
	GVA (stochastic) (stochastic performance profiles)	no	yes	**yes**

Table 1: *When does strategic computation occur? The answer to this question depends on the model of limited computation being used.*

8 Conclusion

Auctions provide efficient and distributed ways of allocating goods and tasks among agents. For rational agents, bidding strategies have been well studied in the game theory literature. However, not all agents are fully rational. Instead they may have computational limitations which curtail their ability to compute valuations for the items being auctioned. This adds another dimension to the agents' strategies as they have to determine not only the best bid to submit, but how to use their computational resources in order to determine their valuations and gain information about the valuations of the other agents participating in the auction.

We presented a model of bounded rationality where agents incur cost for each computation step. We showed that in most standard auction settings, in equilibrium agents are willing to use their limited deliberation resources in order to compute on valuation problems of competitors.

We compared results from the costly computation model with those from a free but limited model. Equilibrium strategies differed, which leads to the conclusion that different models can change the equilibrium outcomes substantially. The classical tools for analyzing auctions, i.e., the Revelation Principle and the Revenue Equivalence Theorem, are no longer directly applicable in the limited computation setting. A reinvestigation of modified analysis tools may be required for settings with bounded rational agents.

Future work in this area includes a full analysis of bidding and deliberation strategies under other models of bounded rationality. For example, in this paper it is assumed that agents have the possibility of skipping deliberation steps, and thus avoiding accumulating further cost. This skipping may be disallowed by an auctioneer who also controls the CPU that the software (bidding) agents run on. If cost is incurred at every step in the auction, agents may have incentives to try and end an ascending auction early. Design of different auction protocols which take into account agents' limitations and lead to as Pareto efficient as possible outcomes is another important research area that naturally stems from our paradigm of modeling deliberation actions as part of each agent's strategy.

References

[1] Eric B Baum and Warren D Smith. A Bayesian approach to relevance in game playing. *Artificial Intelligence*, 97(1–2):195–242, 1997.

[2] Mark Boddy and Thomas Dean. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67:245–285, 1994.

[3] E H Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

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

[5] Eric Hansen and Shlomo Zilberstein. Monitoring and control of anytime algorithms: A dynamic programming approach. *Artificial Intelligence*, 126:139–157, 2001.

[6] Eric J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of Third Workshop on Uncertainty in Artificial Intelligence*, pages 429–444, Seattle, Washington, July 1987.

[7] Philippe Jehiel. Limited horizon forecast in repeated alternate games. *Journal of Economic Theory*, 67:497–519, 1995.

[8] Noa Kfir-Dahav, Dov Monderer, and Moshe Tennenholtz. Mechanism design for resource bounded agents. In *ICMAS 2000*, 2000.

[9] Daphne Koller, Nimrod Megiddo, and Bernhard von Stengel. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, 14(2):247–259, 1996.

[10] Kate Larson and Tuomas Sandholm. Computationally limited agents in auctions. In *Workshop on Agent–based Approaches to B2B, Autonomous Agents*, Montreal, 2001.

[11] Kate Larson and Tuomas W Sandholm. Deliberation in equilibrium: Bargaining in computationally complex problems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 48–55, Austin, 2000.

[12] Daniel Lehmann, Lidian Ita O'Callaghan, and Yoav Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 96–102, Denver, CO, November 1999.

[13] Paul Milgrom. Auctions and bidding: A primer. *Journal of Economic Perspectives*, 3(3):3–22, 1989.

[14] S Minton, M D Johnston, A B Philips, and P Laird. Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58(1–3):161–205, 1992.

[15] Noam Nisan and Amir Ronen. Computationally feasible VCG mechanisms. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 242–252, Minneapolis, MN, 2000.

[16] 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, 1999.

[17] Ariel Rubinstein. *Modeling Bounded Rationality*. MIT Press, 1998.

[18] Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial Intelligence*, 49:361–395, 1991.

[19] Tuomas Sandholm, Subash Suri, Andrew Gilpin, and David Levine. Cabob,: A fast optimal algorithm for combinatorial auctions. In *International Joint Conference on Artificial Intelligence*, Seattle, 2001.

[20] Tuomas W Sandholm. eMediator: A next generation electronic commerce server. In *Proceedings of the Fourth International Conference on Autonomous Agents (AGENTS)*, pages 73–96, Barcelona, Spain, June 2000.

[21] Tuomas W Sandholm. Issues in computational Vickrey auctions. *International Journal of Electronic Commerce*, 4(3):107–129, 2000.

[22] Tuomas W Sandholm and Victor R Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1):99–137, 1997.

[23] Herbert A Simon. A behavorial model of rational choice. *Quarterly Journal of Economics*, 69:99–118, 1955.

[24] Shlomo Zilberstein and Stuart Russell. Optimal composition of real-time systems. *Artificial Intelligence*, 82(1–2):181–213, 1996.