Miscomputing Ratio: Social Cost of Selfish Computing

Kate Larson
Computer Science Department
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213, USA
klarson@cs.cmu.edu

Tuomas Sandholm
Computer Science Department
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213, USA
sandholm@cs.cmu.edu

ABSTRACT

Auctions are useful mechanism for allocating items (goods, tasks, resources, etc.) in multiagent systems. The bulk of auction theory assumes that the bidders' valuations for items are given *a priori*. However, in many applications the bidders need to expend significant computational effort to determine their valuations. We introduce a way of measuring the negative impact of agents choosing computing strategies selfishly. Our *miscomputing ratio* isolates the effect of selfish computing from that of selfish bidding. We present a Bayes-Nash equilibrium analysis of a Vickrey auction where the bidders' strategies include computational actions. This equilibrium analysis allows us to predict the overhead caused by miscomputing, as measured by the miscomputing ratio. We show that in some situations, the outcome can be arbitrarily far from optimal. However, by carefully designing cost functions for agents, it is possible to provide incentives for bidders to choose computing policies that result in optimal social welfare.

Categories and Subject Descriptors: 1.2.11 Multiagent systems
General Terms: Design, Theory
Keywords: Auctions, Resource-bounded agents

1. INTRODUCTION

Auctions are useful mechanisms for allocating items (goods, tasks, resources, etc.) in multiagent systems. The bulk of auction theory assumes that the bidders' valuations for items are given *a priori*. In many applications, however, the bidders need to expend significant effort to determine their valuations. This is the case, for example, when the bidders can gather information [13, 1] or when the bidders have the pertinent information in hand, but evaluating it is complex.

For agents with restrictions on their computational resources, valuation determination can be difficult. Agents must decide how much of their resources to use on their valuation problems. Decision making under settings of bounded resources is challenging even for single agents and the field of AI has long searched for useful techniques for coping with restricted resources (see, for example [17, 15, 3]). For self-interested agents interacting in multiagent systems, the problem of how to use their resources is made even more complex as agents must also take into account the (deliberation and other) actions of other agents in the system. A fully normative deliberation control method coupled with game theoretic analysis is required in order to guarantee that agents' optimal strategies result in desirable outcomes.

A normative deliberation control model of how additional work (e.g., computing) refines valuations was recently introduced [8, 7]. The authors analyzed auctions strategically, where each agent's strategy included both computing and bidding. They found that for certain auctions, properties such as incentive compatibility cease to hold if agents explicitly deliberate to determine valuations. Instead agents strategize and counterspeculate, sometimes using computing to (partially) determine opponents' valuations. It was conjectured that such strategic computing may lead to suboptimal social welfare.

In this paper we introduce a way of measuring the negative impact of agents choosing computing strategies selfishly. Our *miscomputing ratio* isolates the effect of selfish computing from that of selfish bidding. We provide a game-theoretic analysis that allows us to predict the overhead, as measured by our miscomputing ratio. If we only focus on the social welfare of the bidding agents, we show that under both limited and costly computing, the outcome can be arbitrarily far worse than in the case where computations are coordinated. However, by carefully designing computing cost functions, it is possible to provide appropriate incentives for bidders to choose computing policies that result in the optimal social welfare. If the center (auctioneer) is included in the social welfare measure, then cost functions are not required to obtain the optimal social welfare, this can be achieved by enforcing deadlines on agents. Furthermore, if agents' cost functions are exogenous, we can predict what the negative impact of miscomputing will be.

The paper is organized as follows. In Section 2 the auction model and deliberation model are described. Section 3 introduces the miscomputing ratio. Section 4 includes an equilibrium analysis of computational agents in a Vickrey auction, and a discussion of the role of cost functions and how they should be imposed. We conclude with related work and a summary of the paper.

2. THE MODEL

In this section we describe the model. We first review relevant game theoretic solution concepts and the model of deliberation control. We then describe how the deliberation actions of agents can be incorporated into a standard auction setting, and mention some issues that arise.

AAMAS'03, July 14–18, 2003, Melbourne, Australia

2.1 Concepts from Game Theory

A game has a set of agents, I, and a set of outcomes O. Each agent has a set of strategies from which it chooses a strategy to use. A strategy of agent i, s_i, is a contingency plan that determines what action the agent will take at any given point in the game. A strategy profile, $s = (s_1, s_2, \ldots, s_{|I|})$, is a vector specifying one strategy for each agent in the game. We use the notation $s = (s_i, s_{-i})$ to denote a strategy profile where the strategy of agent i is s_i and $s_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1} \ldots, s_{|I|})$. The strategy profile determines how the game is to be played, and thus determines an outcome $o(s) \in O$. Each agent has a utility function, $u_i : O \mapsto \mathbb{R}$. Each agent tries to choose a strategy which leads to an outcome that maximizes its own utility.

Noncooperative game theory is interested in finding stable points in the space of strategy profiles. These stable points are the equilibria of the game. There are many types of equilibria, the two most common being *dominant strategy equilibria* and *Nash equilibria*.

A strategy is said to be dominant is said to be dominant if it is a player's strictly best strategy against any strategies that other agents may follow.

Definition 1 *Agent i's strategy, s_i^* is a dominant strategy is*

$$\forall s_{-i} \forall s_i' \neq s_i^* u_i(o(s_i^*, s_{-i})) > u_i(o(s_i', s_{-i})).$$

If each agent's strategy in a strategy profile is the agent's dominant strategy, then the strategy profile is in a dominant strategy equilibrium.

Agents will often not have dominant strategies and so dominant strategy equilibria do not always exist. Instead a different notion of equilibrium is often used, that of the Nash equilibrium.

Definition 2 *A strategy profile s^* is a* Nash equilibrium *if no agent has incentive to deviate from its strategy given that the other agents do not deviate. Formally,*

$$\forall i \; \forall s_i' \; u_i(o(s_i^*, s_{-i}^*)) \geq u_i(o(s_i', s_{-i}^*)).$$

The Nash equilibrium solution concept assumes that agents know everything about all other agents in the game. Often it is the case that an agent does not know the preference of another agent. These situations are modeled as *Bayesian games*. Each agent's preferences are determined by the realization of a random variable. The agent, i, will be the only one who knows the realization of the random variable (its type, θ_i), however, the probability distribution from which θ_i was drawn is common knowledge. A pure strategy for agent i in a Bayesian game is a function $s_i(\theta_i)$ that gives the player's strategy choice for each possible type θ_i. Agent i's utility if it has type θ_i and $s = (s_1(\cdot), \ldots, s_{|I|}(\cdot))$ is $\overline{u_i}(o(s)) = E_{(\theta_1, \ldots, \theta_{|I|})}[u_i(o(s_1(\theta_1), \ldots, s_{|I|}(\theta_{|I|}), \theta_i)]$. A Nash equilibrium in a Bayesian game is called a *Bayes-Nash equilibrium*.

Definition 3 *A strategy profile s^* is a* Bayes-Nash equilibrium *if no agent has incentive to deviate given that no other agent deviates. That is*

$$\forall i \; \forall s_i' \; \overline{u_i}(o(s_i^*, s_{-i}^*)) \geq \overline{u_i}(o(s_i', s_{-i}^*)).$$

While we focus our attention to outcomes that are obtained by agents playing in equilibrium, there are some equilibrium outcomes that are more desirable than others. One common measure for comparing outcomes is Pareto efficiency. It is a desirable measure in that it does not require cardinal utility comparisons across agents.

Definition 4 *An outcome, o, is* Pareto efficient *if there exists no other outcome o' such that some agent has higher utility in o' than in o, and no agent has lower utility. Formally,*

$$\nexists o' s.t. [\forall i \; u_i(o') \geq u_i(o) \text{ and } \exists j \; u_j(o') > u_j(o)].$$

Another measure that is commonly used is *social welfare*. It allows prioritizing one Pareto optimal outcome over another, but it does require cardinal utility comparison across agents.

Definition 5 *The* social welfare *of outcome $o \in O$ is*

$$SW(o) = \sum_{i \in I} u_i(o).$$

Equilibrium play does not always optimize social welfare. A classic example of this is the Prisoner's Dilemma game where the dominant strategy equilibrium leads to an outcome with lowest social welfare.

The definitions used above were for general utility functions. However, in this paper, as is standard when discussing auctions, we will assume that agents all have *quasilinear* utility functions. That is, the utility is agent i is of the form $u_i = v_i - p_i$ where v_i is the value of the outcome of the auction to the agent and p_i is the amount the agent pays for the item.

2.1.1 Auctions

In this paper we study auctions where one good is being sold. There are numerous auction mechanisms, but we choose to work solely with the Vickrey auction. In a Vickrey auction (also known as a second-price sealed-bid auction) each agent can submit a single sealed bid to the auctioneer. The agent with the highest bid is allocated the item, but pays only the amount of the second highest bid. The desirable feature of this auction is that it has a dominant strategy equilibrium. If bidders know their valuations then their dominant strategy is to reveal the valuations truthfully rather than strategically under or over bidding.

2.2 Model of Deliberation

In order to bid sensibly in an auction, agents need to have a valuation for the item being sold. In this paper we focus on settings where agents do not simply know their own valuations, but, instead must allocate computational resources to determine them.

If agents know their own valuations (or are able to determine them with ease) they can execute the equilibrium bidding strategies for rational agents. However, agents often have restrictions on their capabilities for determining valuations. In this paper we are interested in settings where agents must *compute* to determine valuations. Settings where the value of an item depends on how it is used often have this property. For example, in a vehicle routing domain, jobs are auctioned off to delivery companies. The value of a job depends, among other things, how it fits into the current delivery schedule of a company. To determine this involves solving an NP-complete optimization problem.

In many situations it may not be feasible to optimally solve the valuation problem. Instead, one must resort to some form of approximation. In this paper we assume that agents have *anytime algorithms* for approximating valuations [2]. The defining property of an anytime algorithm is that it can be stopped at any point in time to provide a solution, and the quality of the solution improves as more time is allocated to the problem. This allows for a tradeoff between solution quality and time spent on computing.

Since the computational resources of agents may be restricted, agents must make tradeoffs in how to determine their valuations. Alone, anytime algorithms do not provide an adequate tool set. Instead, they need to be paired with a meta-level control procedure that determines how long to run the algorithm, and when to stop and act with the solution obtained. In this paper we assume that agents use a meta-level control procedure called a performance profile tree [6].

Performance profile trees are created from statistics collected from previous runs of the algorithm on valuation problem instances. A tree describes how deliberation (computation) changes the solution to the valuation problem. Figure 1 is an example of a performance profile tree.

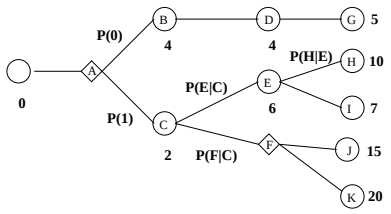

Figure 1: ***An agent's stochastic performance profile tree for a valuation problem. The diamond shaped nodes are random nodes and the round nodes are solution nodes. At random node A, the probability that the random number will be 0 is $P(0)$, and the probability that the random number will be 1 is $P(1)$. At solution node E, the edges are labeled with the probability of reaching each child, given that node E was reached.***

A performance profile tree has two different types of nodes; solution nodes and random nodes. Solution nodes store the solution and other problem features that may be of interest to the agent that the algorithm has computed given a certain amount of computational resources. Random nodes occur whenever a random number is used to chart the path of the algorithm run. The edges of the tree are labeled with the probability that after one more step of computation, the solution returned will be the node found by following the edge.

An agent uses performance profile trees to help in making decisions about how to allocate its resources (for example, computing time). As an agent allots resources (time) to an algorithm, the solutions returned correspond to nodes in the tree. The performance profile tree provides information about how the solution will change with future computation. In particular, if an agent has reached a certain node in the tree, then the agent need only consider solutions in the subtree rooted at that node. The probability of obtaining a solution v', given that the agent has reached a node of solution v is equal to the product of the probabilities of the edges connecting node with solution v to v'. Using this information, agents are able to decide at which point it is best to stop computing.

In order to do game-theoretic analysis one must have a fully normative model for deliberation control because one must take into account the possibility that an agent will use all possible information and resources available to it when determining its best strategy. The performance profile tree is fully normative since it can model all resource usage, algorithms, and solution features that an agent might have available to it. The performance profile tree also allows for optimal conditioning on many parameters, including results of execution so far and on the actual problem instance.

2.2.1 Restrictions on Computational Resources

We model agents' computational restrictions as cost functions. A cost function for agent i is

$$c_i : T \mapsto \mathbb{R}^+$$

where T is the set of resources available to the agent.[1] For example, in many situations time is the resource of interest so that $c_i(t)$ is the cost to agent i for computing t time steps. A common setting is when agents are faced with deadlines. This can be modeled by using cost functions. If D_i is the deadline for agent i and t represents the time the agent has computed then

$$c_i(t) = \begin{cases} 0 & \text{if } t \leq D_i \\ \infty & \text{if } t > D_i \end{cases}$$

2.3 Strategic Computing and Bidding

A strategy for an agent is composed of two interrelated components - the computing component and the bidding component. What an agent bids depends on the solutions that it has obtained for the valuation problems, and the problem an agent decides to compute on depends partially on how it is planning on bidding, and how other agents bid. Figure 2 illustrates this relation.

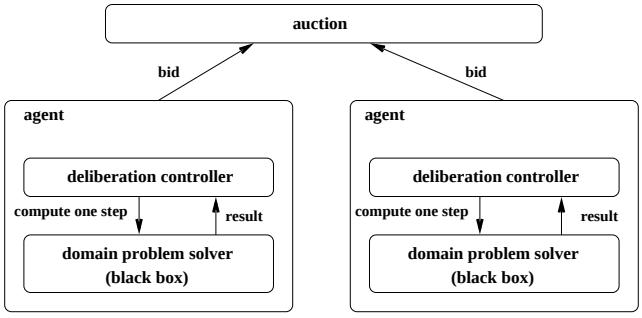

Figure 2: ***An auction with two computationally restricted agents. In order to submit a reasonable bid, each agent needs to first (approximately) compute its valuations for the item that is up for auction.***

A strategy of a computationally restricted agent participating in a Vickrey auction can be formally defined as follows. Let $\{a_1, \ldots, a_I\}$ be the set of computational actions, that is a_i is the action of computing one step on the valuation problem of agent i. A state of computation for agent i at time t, $\theta_i(t) = \langle v_1, \ldots, v_I \rangle$ stores the results the agent has obtained from computing for t time steps. Note that the agent may have computed on its own problem and other agents' valuation problems. An agent's strategy at time t is a mapping from its current state of computation to either a computational action, a_j, or a mapping from its current state of computation to a bid.

The deliberation component can be quite complex, since at each time step there is no restriction as to what problem an agent can compute on. In fact, as mention previously, it may even use some of its resources to compute on the valuation problems of other agents participating in the auction. This is called *strong strategic computing*. It has been shown in earlier work that the restrictions on the agents' resources have a significant impact on what (deliberation) strategies agents may use. In situations where agents are faced with hard deadlines we get the following result.

Theorem 1 *Assume that agents are allowed to compute at no cost, but have deadlines. Then, in a Vickrey auction, the bidders have (weakly) dominant strategies where they only compute on their own valuation problems [7].*

If agents incur a cost while computing, then the deliberation strategies can be quite different.

[1] We can easily generalize the cost functions to be mappings from resource vectors to the real numbers. That is, $c_i : T^{|I|} \mapsto \mathbb{R}$ where $(t_1, \ldots, t_I)$ is a vector that specifies how much resources were allocated to each agent's problem. This allows us to model situations where the cost computing on a problem varies, depending on the agents.

Theorem 2 *Assume that agents have a cost associated with computing. Then, in a Vickrey auction, strong strategic computation can occur in strict Nash equilibrium [8].*

3. THE SOCIAL COST OF SELFISH COMPUTING

Now, a natural question to ask is whether the restrictions on computing resources results in a loss of efficiency. However, efficiency is difficult to compare in such settings. The Vickrey auction is efficient in the sense that it always allocates the item to the bidder with the highest valuation. However, an agent who *might* have been able to obtain the highest valuation via computing may have used its computing resources on a different problem, thus causing a different agent to have the highest valuation and win the auction. This outcome is still efficient *given how agents computed*, but it overlooks the computational issues in an unsatisfying way. This suggests that Pareto efficiency may not always be the right measure to use in the context of computationally bounded agents. Is there an alternative measure?

Instead of looking at efficiency, we propose to use social welfare as the measure. We want to know how letting agents freely choose their own computing strategies impacts the social welfare of the set of all bidders. In particular, we compare the highest achievable social welfare to the lowest social welfare achievable in any Nash equilibrium.

When we determine the highest achievable social welfare we optimistically assume that there is a global controller who imposes each agent's computing strategy (so as to maximize social welfare). The controller has full information about all performance profiles, deadlines, cost functions, and intermediate results of computing, and given this information, specifies exactly how each agent must use its computational resources. In the bidding stage agents are free to bid as they wish, but their goal is still to maximize their own utility, and so they bid truthfully in the Vickrey auction, given the valuations they have obtained under the enforced computing policy.

Definition 6 *Let o^* be the outcome that is reached if the global controller dictates computing policies to all agents, and agents are free to bid in the Vickrey auction.*

On the other extreme, we are interested in what happens when agents are free to choose to follow any computing and bidding strategy. Let NashEq be the set of Nash equilibria in that game. We now define what is meant by the worst-case Nash equilibrium.

Definition 7 *The worst case Nash equilibrium is* $NE = \text{argmin}_{s \in \text{NashEq}} SW(o(s))$.

We use the following ratio to see how much letting agents choose their own computing strategies reduces the social welfare.

Definition 8 *The* miscomputing ratio *is*

$$R = \frac{SW(o^*)}{SW(o(NE))}.$$

This ratio isolates the impact of selfish computing from the traditional strategic bidding behavior in auctions. This is because in both the coordinated and uncoordinated scenario, the agents bid based on self-interest.

We actually study the impact of selfish computing in two slightly different settings. In the first setting we are only interested in the impact on the strategic agents (i.e. the bidding agents). In this case, if I is the set of bidding agents then $SW(o) = \sum_{i \in I} u_i(o)$. We denote the miscomputing ratio in this setting as R_I. In the second setting we study the impact of computation on *all* agents in the game, including the auctioneer. The social welfare of an outcome is computed as $SW(o) = \sum_{i \in I} u_i(o) + u_{\text{auc}}(o)$ where $u_{\text{auc}}(o)$ is either the amount that was paid to the auctioneer by the winning bidder, or else the value the auctioneer has for the item up for auction in the case where no one wins (and so the auctioneer keeps the item). Under this social welfare measure, the miscomputing ratio is denoted by $R_{I \cup \{auc\}}$.

4. RESULTS

In this section we present the results of the paper. We first outline the assumptions made in the paper, and prove a result concerning agents' strategies that is used throughout. We then do a Bayes-Nash equilibrium analysis of the Vickrey auction with computationally restricted agents. Using this equilibrium analysis, it is possible to determine the miscomputing ratio under different conditions. We show that sometimes it is best to just allow all agents to compute freely, while at other times cost functions can be used to motivate all agents to compute "correctly".

4.1 Agents' Nondominated Strategies

Assume there is a set of bidding agents, I, who are competing in a Vickrey auction being run by auctioneer agent, auc. Before a bidding agent can submit a bid, it must first use some of its resources to determine its valuation. For ease of exposition, we assume that each agent i has a deterministic algorithm, v_i, and performance profiles and that these are common knowledge.[2] While this assumption does make the analysis simpler in that there is less uncertainty in what possible values have been computed, agents with deterministic performance profiles can still suffer from miscomputing.

Each agent has to expend some of its resources (time) while computing. We model this expenditure by assuming that each agent i has a cost function, as described in Section 2. The cost function of agent i, c_i, is private. However, the cost function is drawn from some set $\mathcal{C}_i$ by distribution $f_i : \mathcal{C}_i \mapsto [0, 1]$ where f_i is common knowledge.

An agent's utility depends on whether it has computed, what value it has obtained by computing, and whether it has won the auction. If an agent does not compute then it does not have a valuation for the item, and so we state that its utility is 0. Assume an agent computes for t_{own} on its own problem and t_{total} on all problems. If it does not win the auction, then its utility is $-c(t_{\text{total}})$. If it does win the auction, and the second highest bid is b, then its utility is $v(t_{\text{own}}) - c(t_{\text{total}}) - b$.

Given these assumptions, we can now determine agents nondominated strategies.

Theorem 3 *Assume that agent $i \in I$ has a deterministic algorithm v_i and some cost function $c_i(\cdot)$. Then, agent i has only two possible nondominated strategies. It will either not compute at all, or it will compute for t_i^* steps on its own problem where*

$$t_i^* = \arg\max_t \{v_i(t) - c_i(t)\}.$$

If agent i does compute, then it submits a bid equal to $v_i(t_i^*)$.

Proof First, the argument that an agent is best of submitting a bid equal to the value that it has computed is identical to the proof that bidding truthfully is optimal for rational agents. Thus, we omit this

[2]This assumption is not necessary. We can also assume the more general situation where v_i is drawn with distribution $g_i()$ from the set of algorithms V_i, where $g_i()$ is common knowledge.

part of the proof. If an agent does not compute at all, then it can not submit a bid and be allocated the item. We use this as the default situation, and assume that an agent i will have utility $u_i = 0$.

Second, agent i is best off never computing on another agent's valuation problem. Since the algorithms are deterministic and common knowledge, agent i already knows what $v_j(t)$ is for any agent j and any time t. Therefore, by computing on agent j's problem it gains no new knowledge, yet incurs a cost.

Finally, if the agent does compute for t time steps then its utility is

$$u_i = \begin{cases} v_i(t) - c_i(t) - b & \text{if } v_i(t) > b = \max_{j \neq i}\{b_j\} \\ -c_i(t) & \text{otherwise} \end{cases}$$

If agent i won the auction, then it is best off computing so as to maximize its utility, that is computing for t_i^* steps on its problem. If agent i did not win, then it is best off not having computed at all ($u_i = 0$). □

4.2 When Should an Agent Compute?

From Theorem 3 we know that agent i should either compute so as to maximize $v_i(t) - c_i(t)$ or not compute at all. Which strategy it should follow depends on what other bidding agents are doing. Figure 3 illustrates how an agent's utility depends on both its computational actions and the bids of other agents.

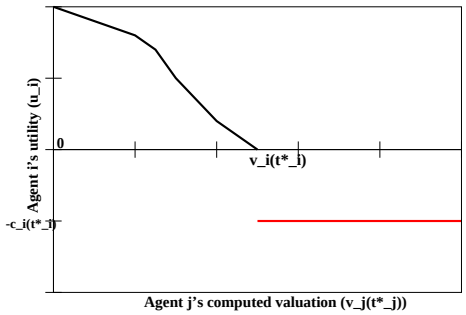

Figure 3: ***Assuming that both agent i and agent j have computed, the utility of agent i is a decreasing function of the computed valuation of agent j. As agent j's computed valuation increases, agent i's utility decreases. Agent i's utility is 0 when $v_j(t_j^*) = v_i(t_i^*)$. When $v_j(t_j^*) \geq v_i(t_i^*)$, the utility of agent i is equal to $-c_i(t_i^*)$.***

So far we have determined how much an agent should compute, if it decides to, but we still do not know under what conditions an agent should decide to compute. If agent i knew every other agents' computed valuations and thus what they would bid, it would be able to optimally decide whether to compute or not. While an agent does not have this information available to it, it does have probabilistic information about the other agents' cost functions. It can use this information to derive a distribution over every other agents' possible computed valuations. Recall that $\mathcal{C}_j$ is the set of possible cost functions for agent j and $f_j(c_j(\cdot))$ is the probability that agent j has cost function $c_j(\cdot)$. The distribution, $\overline{f}_j$, over the valuations that agent j may have computed is

$$\overline{f}_j(x) = \int_{\mathcal{C}_j} f_j(c_j)\chi_{c_j}(x)dc_j$$

where

$$\chi_{c_j}(x) = \begin{cases} 1 & \text{if } x = \max_t v_j(t) - c_j(t) \\ 0 & \text{otherwise.} \end{cases}$$

For ease of presentation we assume that there are just two bidding agents, i and j.[3] Let $s_j(x)$ represent the strategy of agent j if $v_j(t_j^*) = x$. That is

$$s_j(x) = \begin{cases} 1 & \text{if } j \text{ computes when } x = v_j(t_j^*) \\ 0 & \text{if } j \text{ does not compute when } x = v_j(t_j^*) \end{cases}$$

Let $P_j(x)$ be the probability that $s_j(x) = 0$.

The expected utility from computing for agent i on what agent j decides to do. If agent j does not compute, then, by the rules of the Vickrey auction, agent i will win the auction, and will pay nothing for the item. Its utility will be $u_i = v_i(t_i^*) - c_i(t_i^*)$. If agent j does compute, then the utility of agent i depends on whether the computed valuation of agent j is greater than its own. If $v_j(t_j^*) < v_i(t_i^*)$, then the utility of agent i is $u_i = v_i(t_i^*) - c_i(t_i^*) - v_j(t_j^*)$. However, if $v_j(t_j^*) > v_i(t_i^*)$ then $u_i = -c_i(t_i^*)$. This is captured in the following equation:

$$\begin{aligned} u_i \quad = \quad & \underbrace{\int_0^{\infty} P_j(x)\overline{f}_j(x)(v_i(t_i^*) - c_i(t_i^*))dx}_{j \text{ does not compute}} \\ & + \underbrace{\int_0^{v_i(t^*)} (1 - P_j(x))\overline{f}_j(x)(-c_i(t_i^*))dx}_{j \text{ computes and } v_j(t_j^*) \geq v_i(t_i^*)} \\ & + \underbrace{\int_{v_i(t_i^*)}^{\infty} (1 - P_j(x))\overline{f}_j(x)(v_i(t_i^*) - c_i(t_i^*) - x)dx}_{j \text{ computes and } v_j(t_j^*) < v_i(t_i^*)} \end{aligned}$$

Agent i will only compute if the above equation is greater than its utility from not computing ($u_i = 0$). Therefore, agent i will only compute under the condition

$$\begin{aligned} c_i(t_i^*) \quad \leq \quad & v_i(t_i^*)\left[1 + \int_{v_i(t_i^*)}^{\infty} P_j(x)\overline{f}_j(x)dx\right] \\ & - \int_0^{v_i(t_i^*)} (1 - P_j(x))\overline{f}_j(x)xdx \end{aligned}$$

This is a cutoff equilibrium. Agent i will compute only when its cost for computing is below a certain threshold.

4.3 Examples

With the analysis in the previous section, we are able to determine when an agent should compute and when it should not.

4.3.1 *Free Computation with Deadlines*

Assume that all agents have cost functions of the following form

$$c_i(t) = \begin{cases} 0 & \text{for } t \leq D_i \\ \infty & t > D_i. \end{cases}$$

This corresponds to the situation where agents have free computation but must stop computing by some deadline. We will assume that the uncertainty between the agents about the cost functions is

[3]The multiple agent setting is not conceptually more difficult. The techniques and logic used to find the equilibrium is the same as the two agent setting. However, one must keep track of multiple probability distributions which makes the notation cumbersome. For this reason, we present the two agent case.

caused by having the deadlines be private information. For each agent i, $t_i^* = D_i$ since $D_i = \arg\max_t\{v_i(t) - c_i(t)\}$. Clearly

$$v_i(D_i)\left[1 + \int_{v_i(D_i)}^{\infty} P_j(x)\overline{f}_j(x)dx\right] \geq v_i(D_i).$$

Since

$$\int_0^{v_i(D_i)}(1 - P_j(x))x\overline{f}_j(x)dx \leq v_i(D_i)$$

it is always the case that for $t_i^* = D_i$

$$\begin{aligned} 0 = c_i(t_i^*) \quad &\leq \quad v_i(t_i^*)\left[1 + \int_{v_i(t_i^*)}^{\infty} P_j(x)\overline{f}_j(x)dx\right] \\ &\quad - \int_0^{v_i(t_i^*)}(1 - P_j(x))\overline{f}_j(x)xdx. \end{aligned}$$

That is, agent i will compute on its own problem until its deadline D_i.

4.3.2 Constant Cost Function

Assume that agents i and j share an algorithm, v, such that for all t, $v(t) = V$. Assume that the agents have constant cost functions, c_i, c_j such that for all t, $c_i(t) = K_i$ and $c_j(t) = K_j$, where K_i, K_j are drawn uniformly from the interval $[0, K]$ for some constant K. Each agent knows its own cost function, but only knows that its competitor's cost function is drawn uniformly from $[0, K]$.

If agent i decides to compute on its problem, its utility depends on whether or not agent j also decided to compute on its problem. Assuming that agent i computed, its utility is

$$u_i = \begin{cases} V - K_i & \text{if agent } j \text{ did not compute} \\ -K_i & \text{if agent } j \text{ did compute} \end{cases}$$

since, if both agents computed then the item could be allocated to either of them but they would have to pay V. In equilibrium, agent i will only compute when its expected utility from computing is greater than not computing. That is

$$0 \leq \int_0^K P_j(x)(V - K_i)dx + \int_0^K (1 - P_j(x))(-K_i)dx$$

As before, this is a cutoff equilibrium. Let $\hat{K}_i$ and $\hat{K}_j$ be the costs at which each agent switches from a strategy involving computing to one where it does not compute. Then, in equilibrium,

$$\hat{K}_i = \int_0^K \frac{\hat{K}_j}{K}(V - \hat{K}_i)d\hat{K}_j + \int_0^K (1 - \frac{\hat{K}_j}{K})(-\hat{K}_i)d\hat{K}_j.$$

This reduces to

$$\hat{K}_i = \frac{VK}{2(K+1)},$$

that is, agent i will compute whenever its cost of computing of less than $VK/2(K+1)$. The same cutoff equilibrium holds for agent j also. Figure 4 displays how the cutoff changes as K approaches V.

4.4 The Miscomputing Ratio

Given the tool set derived above, it is now possible to estimate what the miscomputing ratio is for any Vickrey auction. By using the Bayes-Nash equilibrium we can figure out, in expectation, what the miscomputing ratio will be given information about agents cost functions.

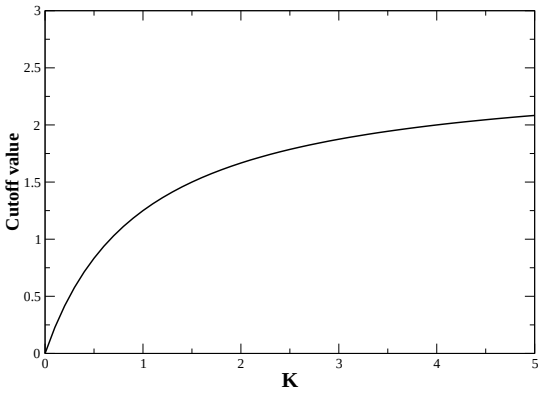

Figure 4: ***The cutoff values for agent i as a function of K. For each value of K, if agent i's cost of computation falls below the line then it will compute. Otherwise, it does not. In this example, $V = 5$.***

First, we have learned that if agents have free but limited computation, they will compute on their own problems only. What is the miscomputing ratio is this setting?

Theorem 4 *Let I be the set of bidding agents in a Vickrey auction. Assume that $|I| \geq 2$, and that each agent, i, has free computation and deadline D_i. Then, if the auctioneer is included in the social welfare measure, the miscomputing ratio is $R_{I\cup\{auc\}} = 1$.*

Proof To maximize social welfare, the global controller would select agent i such that $v_i(D_i) = \max_{j\in I} v_j(D_j)$, and only allow this agent to compute for D_i time steps. All other agents would be forbidden to compute. Agent i would submit a bid equal to $v_i(D_i)$ and all other agents would submit a bid equal to 0. The social welfare of this outcome, o^*, is $SW(o^*) = v_i(D_i)$. In equilibrium, as we have seen in, every agent would compute on its own problem until it reached its own deadline. Each agent would then submit a bid of an amount equal to its computed valuation. Agent i where $v_i(D_i) = \max_{j\in I} v_j(D_j)$ would be allocated the item. However, the price agent i would have to pay is equal to

$$v_k(D_k) = \max_{j\in I\setminus\{i\}} v_j(D_j).$$

The auctioneer's utility from this outcome is $u_{auc} = v_k(D_k)$. The social welfare is

$$SW(NE(o)) = (v_i(D_i) - v_k(D_k)) + v_k(D_k) + 0.$$

Therefore, the miscomputing ratio is

$$R_{I\cup\{auc\}} = \frac{v_i(D_i)}{(v_i(D_i) - v_k(D_k)) + v_k(D_k)} = 1.$$

□

What happens if we do not include the auctioneer in the computation of the ratio? It turns out that with limited computing, the miscomputing ratio can be arbitrarily bad.

Theorem 5 *Let I be the set of bidders in a Vickrey auction ($|I| \geq 2$). Assume that each bidder i has free but limited computing with deadline D_i. Then, the miscomputing ratio R_I can be infinity.*

Proof

Each agent has a dominant strategy which is to deliberate only on its own valuation problem until its deadline and to submit a bid equal to the valuation that it has obtained. That is, agent i submits

a bid of $v_i(D_i)$. Without loss of generality, assume that $v_1(D_1) \geq v_2(D_2) \geq v_j(D_j)$ for all $j \neq 1, 2$. In equilibrium, agent 1 will win the auction and pay an amount of $v_2(T)$. Therefore, agent 1's utility is $u_1 = v_1(D_1) - v_2(D_2)$. Set $u_1 = \epsilon$. The utility for all other agents is $u_i = 0$ for $i \neq 1$. Therefore,

$$SW(o(\text{NE})) = \sum_{j=1}^{n} u_j = \epsilon.$$

In order to maximize social welfare, the global controller would prohibit all agents expect for agent 1 from deliberating. Agent 1 would compute on its valuation problem until time D_1 and submit a bid of $v_1(D_1)$ while all other agents would submit a bid of 0. Agent 1 would win the item and pay an amount of 0. The utility for agent 1 is $u_1 = v_1(D_1) - 0 = v_1(D_1)$, while $u_i = 0$ for all $i \neq 1$. Therefore

$$SW(o^*) = \sum_{j=1}^{n} u_j = v_1.$$

The miscomputing ratio, R_I, is

$$R_I = \frac{SW(o^*)}{SW(o(\text{NE}))} = \frac{v_1(D_1)}{\epsilon}.$$

As $\epsilon \rightarrow 0$ (that is, as the difference between the highest and second highest valuations decreases), $R \rightarrow \infty$. □

This is a negative result. Allowing agents to choose their computing strategies leads to an outcome that can be arbitrarily far from optimal.

Prior literature has shown that in Vickrey auctions, computationally limited agents have no incentive to use strong strategic computing (i.e., they do not counterspeculate each other) while agents with costly computing do [8, 16]. This suggests that if there is a system designer who can control how the agents' computational capabilities are restricted, the designer should rather impose limits than costs.

However, it turns out that computing costs can be adjusted so that the optimal miscomputing ratio ($R = 1$) is reached. This would mean that charging for computing is at least as desirable as imposing limits.

Theorem 6 *Computing cost functions can be used to motivate bidders to choose strategies that maximize social welfare.*

Proof Consider the following example. Let there be 2 agents, agent 1 and agent 2, each with a deterministic performance profile. Assume that each agent has free but limited computing resources, and that the deadlines are D_1 and D_2. Each agent has a dominant strategy, which is to deliberate on their own problem and submit a bid of $v_i(D_i)$. Assume that $v_1(D_1) > v_2(D_2)$. The equilibrium outcome is to award the item to agent 1 and have agent 1 pay an amount $v_2(D_2)$. Agent 1's utility is then $u_1 = v_1(D_1) - v_2(D_2)$ while agent 2's utility is $u_2 = 0$. To maximize social welfare the global controller would forbid agent 2 to deliberate, and thus agent 1 could get the item and need not pay anything. The maximum social welfare would be $u_1 = v_1(D_1)$. Therefore

$$R = \frac{v_1(D_1)}{v_1(D_1) - v_2(D_2)}$$

Next, consider the case where a simple cost function is introduced. Define

$$c_i(t) = \begin{cases} k \text{ if } t \leq D_i; \\ \infty \text{ if } t > D_i; \end{cases}$$

for some constant k, $0 < k \leq v_2(D_2) \leq v_1(D_1)$. Any strategy that involves deliberating on the other agent's valuation problem is dominated as the computing action incur a cost without improving the agent's overall utility. Thus, the remaining strategies are for the agents to compute only on their own valuation problems until the cost becomes too high, or not to compute at all. The game is represented in normal form in Table 1.

	compute	no
compute	$v_1(D_1) - v_2(D_2) - k, -k$	$v_1(D_1) - k$, 0
no	0, $v_2(D_2) - k$	0,0

Table 1: ***Normal form game representation of the Vickrey auction where agent i has cost k for computing until D_i, and infinite cost if it computes for more than D_i. Agent 1 is the row player and agent 2 is the column player. Each agent would submit a bid that is equal to its computed valuation minus the cost spent to obtain the valuation.***

The sole Nash equilibrium is for agent 1 to compute and submit a bid of $v_1(D_1)$ and for agent 2 to not compute. The global controller trying to maximize the social welfare would force each agent to also follow those strategies. Therefore

$$R_I = \frac{v_1(D_1) - k}{v_1(D_1) - k} = 1.$$

□

In the proof the constant k can be made arbitrarily close to zero. Therefore, the maximum social welfare generated by the global controller in the costly computing setting and be made arbitrarily close to the maximum social welfare obtainable if computing resources are free.

The results suggest that in many settings cost functions can be a useful tool for providing appropriate incentives for agents to compute in such a way so as to maximize social welfare. To provide incentives, the mechanism designer is not required to know anything about the agents performance profiles. Instead, just enforcing a small, constant cost function can possibly provide the correct incentives.

There are several present day real life scenarios where designers have the capability to enforce cost functions. For example, many supercomputer centers operate by having groups pay for the number of computer hours that they use. This is clearly modeled by the costly computation formalization presented in this paper. Another example is mobile agent platforms. The owner of the platform can impose whatever restrictions it wants on the agents using the platform, including, potentially, cost functions.

5. RELATED RESEARCH

In auctions, computational limitations have been discussed both as they pertain to bidding agents and as they pertain to running the auction (the mechanism). For bounded–rational bidding agents, Sandholm noted that under a model of costly computing, the dominant strategy property of Vickrey auctions fails to hold [16]. Instead, an agent's best computing action can depend on the other agents. In recent work, auction settings where agents have hard valuation problems have been studied [8, 7, 12]. Parkes presented auction design as a way to simplify the meta–deliberation problems of the agent, with the goal of providing incentives for the "right" agents to deliberate for the "right" amount of time [12]. Recently Larson and Sandholm have been working on incorporating computing actions into agents' bidding strategies using a normative model

of deliberation control and have focused on equilibrium analysis of different auction settings under different deliberation limitations [7, 8]. While we borrow the deliberation model from Larson and Sandholm, this paper addresses a different question than previous work. They investigate the impact of restricted computing capabilities on agents' strategies. We look, instead, at what the impact is at a system-wide level, present a measure for comparing overhead in different settings, and ask whether it is possible to place certain bounds on the overhead added by having resource-bounded agents.

There has also been recent work on computationally limited mechanisms. In particular, research has focused on the generalized Vickrey auction and has investigated ways of introducing approximate algorithms or using heuristics to compute outcomes without loosing incentive compatibility [10, 4, 9]. Our work is different in that it is focused on settings where the *agents* are computationally limited.

Koutsoupias and Papadimitriou [5] first proposed the concept of worst-case Nash equilibrium to measure the *price of anarchy* [11]. They focused on a network setting where agents must decide how much traffic to send along paths in the network. The agents did not have computational limitations. Roughgarden and Tardos studied a different model of network routing using the same measure as Koutsoupias and Papadimitriou and obtained tight bounds as to how far from optimal network usage the agents would be, if allowed to send traffic as they wished [14].

6. CONCLUSIONS

Auctions are useful mechanism for allocating items (goods, tasks, resources, etc.) in multiagent systems. The bulk of auction theory assumes that the bidders' valuations for items are given *a priori*. In many applications, however, the bidders need to expend significant effort to determine their valuations. In this paper we studied computational bidder agents that can refine their valuations using computation. We used a fully normative model of deliberation control for each agent to determine how much computing the agent invests on its own, and others', valuation problems.

We focused on the Vickrey auction where bidding truthfully is a dominant strategy in the classical model. In this paper we introduced a way of measuring the negative impact of agents choosing computing strategies selfishly. Our *miscomputing ratio* compares the social welfare obtainable if a global controller enforces computing policies designed to maximize social welfare (but does not impose bidding strategies), to the social welfare that is obtained in the worst Nash equilibrium. This measure isolates the effect of selfish computing from that of selfish bidding.

We presented a Bayes-Nash equilibrium analysis of a Vickrey auction where the bidders' strategies include deliberation actions. The equilibrium showed how each agent's cost of computing determines the agent's strategy. The model allowed us to predict the overhead caused by miscomputing. It also allowed for the *design* of cost functions for computing. When including the auctioneer in the welfare measure, free computing with a deadline is an optimal way to control the cost of computing. If the auctioneer is not included in the ratio then the outcome can be arbitrarily far worse than in the case where computations are coordinated. However, by the careful design of cost functions, it is possible to provide appropriate incentives for bidders to choose computing policies that result in the optimal social welfare. Unlike earlier results, this suggest that if a system designer can choose how to restrict the agents' computing, imposing costs instead of limits may be the right approach.

Acknowledgments

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

7. REFERENCES

[1] Dirk Bergemann and Juuso Välimäki. Information acquisition and efficient mechanism design. *Econometrica*, 70:1007–1034, 2002.

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

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

[4] 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)*, 2000.

[5] Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *Symposium on Theoretical Aspects in Computer Science*, 1999.

[6] Kate Larson and Tuomas Sandholm. Bargaining with limited computation: Deliberation equilibrium. *Artificial Intelligence*, 132(2):183–217, 2001.

[7] Kate Larson and Tuomas Sandholm. Computationally limited agents in auctions. In *AGENTS-01 Workshop of Agents for B2B*, pages 27–34, Montreal, Canada, May 2001.

[8] Kate Larson and Tuomas Sandholm. Costly valuation computation in auctions. In *Theoretical Aspects of Rationality and Knowledge (TARK VIII)*, pages 169–182, Sienna, Italy, July 2001.

[9] Daniel Lehmann, Lidian Ita O'Callahan, and Yoav Shoham. Truth revelation in rapid, aproximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.

[10] 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.

[11] Christos Papadimitriou. Algorithms, games and the Internet. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, pages 749–253, 2001.

[12] 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.

[13] Nicola Perisco. Information acquisition in auctions. *Econometrica*, 68(1):135–148, January 2000.

[14] Tim Roughgarden and Éva Tardos. How bad is selfish routing? In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, November 2000.

[15] Stuart Russell and Eric Wefald. *Do the right thing: Studies in Limited Rationality*. The MIT Press, 1991.

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

[17] Herbert A Simon. *Models of bounded rationality*, volume 2. MIT Press, 1982.