

# Bidders with Hard Valuation Problems\*

**Kate Larson**

**Tuomas Sandholm**

*Computer Science Department*

*Carnegie Mellon University*

*5000 Forbes Ave*

*Pittsburgh, PA 15213*

KLARSON@CS.CMU.EDU

SANDHOLM@CS.CMU.EDU

## Abstract

We investigate deliberation and bidding strategies of agents who are limited in their deliberation capabilities by either deadlines or cost and who are participating in auctions. The agents do not *a priori* know their valuations for the items being auctioned. Instead they devote computing resources to compute their valuations. We present a normative model of bounded rationality where deliberation actions of agents are incorporated into strategies and deliberation 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 computing model of bounded rationality with the model where computing is free but limited. For the English and Vickrey auctions the equilibrium strategies are substantially different in that in free but limited computing agents will not compute on each others' problems, but under the costly model there exist instances where agents may compute on each others' problems in equilibrium. 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

It is an unavoidable fact that most systems will have bounded resources. There are restrictions on memory, space and computing cycles; Tasks have to be completed by deadlines; The costs of obtaining solutions may be prohibitively expensive. In such settings, perfect rationality does not always make sense. Perfect rationality, as used by economists, is focused on what actions agents make as opposed to how the agents decided to choose these actions. For example, the perfect rationality paradigm does not make a distinction between an agent exhibiting intelligent behavior that does the right thing by some means, and intelligent behavior that is the result of intelligent reasoning. When there are bounded resources, it is not fair to judge an agent irrational if it fails to take the best action. It well may have been behaving rationally *given the resources available to it*.

In multiagent systems the impact of bounded resources has large influences. There has been a move from having multiagent systems with a central designer who controls the behavior of all system components, to having a system designer who can control only the *mechanism* (rules of the game), while allowing each agent to choose their own strategies. The efficiency of the system depends on the agents' strategies. So, to develop a system that leads to desirable social outcomes the designer must ensure that each agent is motivated to behave in the desired way. This can be done using the Nash equilibrium solution concept from game theory (or a refinement); no agent is motivated to deviate from its strategy given that the others do not deviate [27, 24]. The problem is that the equilibrium for rational agents does not generally remain an equilibrium for computationally bounded agents. This leaves a potentially hazardous gap in game theory as well as automated negotiation since agents may no longer be motivated to behave in a desired way.

Decision making under settings of bounded resources is challenging even for single agents. The field of artificial intelligence has long searched for useful techniques for coping with restricted resources. Herbert Simon advocated that agents should forgo perfect rationality in favor of limited, economical reasoning. He

---

\*. This paper includes and extends results found in [19] [21].

claimed that “the global optimization problem is to find the least–cost, or best–return decision, net of computational costs” [41]. Considerable work has focused on developing *normative* models that prescribe how a computationally limited agent *should* behave (see, for example [10, 35, 7]). This is a highly nontrivial undertaking, encompassing numerous fundamental and technical difficulties. As a result most of those methods resort to simplifying assumptions such as myopic deliberation control [37, 2], conditioning the deliberation control on hand-picked features [37], assuming that an algorithm’s future performance can be deterministically predicted using a performance profile [11, 12], assuming that an anytime algorithm’s future performance does not depend on the run on that instance so far [4, 45, 44, 10] or that performance is conditioned on quality so far but not the path [9], or resorting to asymptotic notions of bounded optimality [36].

While such simplifications can be 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. Therefore, a fully normative deliberation control method is required as a basis for analyzing each agent’s best strategy. This paper presents such a fully normative deliberation control method. It takes into account that each agent may use all the information it has available to control its computing, including conditioning on the problem instance and the path of solutions found on the run so far. This paper will discuss how this deliberation control method can be used as a basis for deciding on an agent’s best–response strategy: what deliberation actions and bidding actions the agent should execute at any point in the game.

Game theorists have also realized the significance of computing limitations (see, for example [34]), but the models that address this issue have mostly analyzed how complex it is to compute the rational strategies [15] (rather than the computing impacting the strategies), memory limitations in keeping track of history in repeated games via deterministic finite automata or Turing machines (see, for example, [1, 29, 8]), limited uniform-depth lookahead capability in repeated games [13], or showing that allowing the choice between taking one computing action or not undoes the dominant strategy property in a Vickrey auction [40]. On the other hand, in many multiagent settings the limited rationality stems from the complexity of each agent’s (optimization) problem, a setting which is ubiquitous in practice.

In this paper we investigate deliberation and bidding strategies of agents that have limitations on their computing resources.<sup>1</sup> These limitations take one of two forms. There may be deadlines where agents can compute freely up until a certain point in time, at which point all computing must cease. The second limitation may take the form where agents have unlimited but costly computing resources. They do not *a priori* know their valuations for the items being auctioned. Instead, they must devote computing resources in order to compute their valuations. They can also compute on other agents’ valuation problems so as to gain information about the others’ valuations, allowing for better strategic bidding. 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 computing. We analyze different auction mechanisms and settings, presenting results on whether or not strategic computing occurs in equilibrium.

## 2. Game Theory and Auctions

In this section we provide an overview of some key concepts from game theory, and a description of some common auction mechanisms. We also describe the equilibrium strategies of fully rational agents in the different auctions.

### 2.1 Game Theory

An extensive game is a detailed description of the sequential structure of the decision problems encountered by the players in a strategic situation. A game consists of a set of agents,  $I$  ( $|I| = n$ ), a set of actions available to each agent,  $i$ , a set of histories for each agent  $i$  (sequences of actions taken by agent  $i$ ), and a set

---

1. In the paper the terms computing and deliberation shall be used interchangeably.

of outcomes,  $O$ . Each agent is free to choose which strategy to use where a strategy is a contingency plan that determines what action the agent will take at any given point in the game. A *strategy profile*,  $s = (s_1, \dots, s_n)$ , is a vector that specifies one strategy for each agent  $i$  in the game. We use the notation  $s = (s_i, s_{-i})$  to denote a strategy profile where agent  $i$ 's strategy is  $s_i$  and  $s_{-i} = (s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_n)$ . The strategies in a strategy profile determine how the game is played out, and that determine the outcome,  $o(s) \in O$ . Each agent  $i$  tries to choose a strategy  $s_i$ , so as to maximize its utility, which is given by a utility function  $u_i : O \mapsto \mathbb{R}$ .

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 but in this paper we focus on two of the most common ones: *dominant strategy equilibria* and *Nash equilibria*.

A strategy is said to be dominant if it is an agent's strictly best strategy, against any strategy the other agents might play.

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

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

The strategy is weakly dominant if the inequality is not strict.

If in a strategy profile  $s = (s_1^*, \dots, s_n^*)$ , each strategy  $s_i^*$  is a dominant strategy for agent  $i$ , then the strategy profile  $s$  is a *dominant strategy equilibrium*. Agents do not always have dominant strategies, and so dominant strategy equilibria do not always exist. Instead a different notion of equilibrium is often used, 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 players 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 concept assumes that all agents have full information about the other agents in the game. This may not always hold. Instead, agents may have secret information which influences their strategy choices. The actions an agent takes may provide information to the other agents in the game. These signals (observed actions taken by the agent) can be used by others to update their beliefs about the private information of the agent. A Bayes-Nash equilibrium is a Nash equilibrium where agents have private information and update their beliefs about other agents' private information by applying Bayes Rule after observing signals from the others.

## 2.2 Auctions

Auctions are stylized markets and have been well studied in the game theory and economics literature [25, 43]. The term *auction* is used for settings where there is one seller of items and multiple buyers.<sup>2</sup> In particular, auctions are bidding mechanisms that are described by a set of rules which specify how a winner is determined, and how much the winner should pay. Auctions can be classified in many different ways. As is commonly done, in this paper we focus solely on auction categories where agents have private values, are risk neutral and have quasilinear utility functions. That is, the value of an item to a bidder depends only on that bidder's own preferences (private value), and if an agent wins an item in the auction, the agent's utility is equal to its own value for the item minus the amount that it must pay ( $u_i = v_i - p_i$ ). There are many different types of auction mechanisms that have these properties. In this paper we will study standard single item and multiple item auctions.

---

2. A reverse auction has one buyer and multiple sellers. In this paper the focus is on auctions, though the techniques presented work equally well for reverse auctions.

### 2.2.1 SINGLE-ITEM AUCTIONS

As the name implies, in single item auctions there is only one item for sale. Agents place bids on the item and the auctioneer determines who gets the item and what amount should be paid. There are three commonly studied single item auctions; Ascending, First-price-sealed-bid, and Vickrey. In this paper we will study these three auctions.

In an *ascending auction* the auctioneer raises the price of the item. At any point in time a bidder can decide to withdraw from the auction.<sup>3</sup> It does this by making an announcement to the auctioneer. Once a bidder has left the auction, it is not allowed to reenter. For rational agents there is a dominant strategy. An agent should stay in the auction until the price is equal to its value. As soon as the price rises above an agent's value, then that agent should withdraw.

each bidder is free to raise its bid. When no bidder is willing to increase the bid further, the auction ends with the item being allocated to the agent with the highest bid. That agent pays the amount of its bid. For rational agents there is an optimal best response bidding strategy. Agents keep bidding some small amount  $\epsilon$  more than the previous high bid until they reach their valuation. They stop bidding at that point. This is a (weakly) dominant strategy [17].

In a *first-price sealed-bid auction* each agent submits one bid without knowing the other agents' bids. The highest bidder wins the item and pays the amount of her bid. There is no dominant bidding strategy for rational agents. An optimal strategy depends on the bids of the other agents [17].

The third auction type is the *Vickrey auction* or *second-price sealed-bid auction*. Each bidder submits one bid without knowing what the others' bid. The highest bidder wins the item but pays the amount of the second highest bid. For rational agents there is a (weakly) dominant strategy which is for each agent to bid its true valuation [17].

### 2.2.2 MULTI-ITEM AUCTIONS

In auctions where multiple distinguishable items are sold, bidding strategies for agents can be complex. A bidder's valuation for a combination of items might not be the sum of the individual items' valuations. It may be greater, smaller, or the same. In traditional auction formats where items are auctioned separately, in order to decide how much to bid on an item, an agent needs to estimate which other items it will receive in the other auctions. This can lead to inefficient allocations where bidders do not get the combinations they want or else get combinations that they do not want [40].

*Combinatorial auctions* can be used to overcome these deficiencies. In a combinatorial auction, bidders may submit bids on combinations of items which allows the bidders to express complementarities between items. Based on the bids on the combinations of items, or *bundles*, the goods are *allocated* to the agents. Let  $X = \{x_1, \dots, x_m\}$  be a set of items. A bundle is a subset of the items, for example,  $\{x_1\}$  or  $\{x_1, x_m\}$ . An allocation of items among a set,  $I$ , of agents is  $Y = (y_1, \dots, y_{|I|})$  where  $y_i \subseteq X$ ,  $\cup_{i=1}^{|I|} y_i \subseteq X$  and  $y_i \cap y_j = \emptyset$  for  $i \neq j$ . The *generalized Vickrey auction* (GVA) is a combinatorial auction where the payments are structured so that each bidder's dominant strategy is to bid truthfully. It is an application of the Clarke tax mechanism to auctions [5].

The generalized Vickrey auction (GVA) works in the following manner.

1. Each agent declares a valuation function. So  $v_i(y_i)$  is agent  $i$ 's valuation for allocation  $Y$  where it is given  $y_i$ .<sup>4</sup>
2. The GVA chooses an optimal allocation  $Y^* = (y_1^*, \dots, y_{|I|}^*)$  that maximizes the sum of all the agents' declared valuations.

---

3. This auction is sometimes called a Japanese auction. It differs from the standard English auction because the price is controlled by the auctioneer, not the agents. This means that agents can not "jump bid" or need to be concerned with other strategic behavior.

4. This paper focuses on combinatorial auctions where the value of an agent depends only on the items that it is allocated. It does not care what items are allocated to other agents. In allocation auction, however, an agent's value can depend on both the items allocated to it, and the items allocated to others.

Auction	Dominant strategy equilibrium?
First-price-sealed-bid	no
Ascending	yes
Vickrey	yes
GVA	yes

Table 1: For rational agents the type of equilibrium outcome depends on the auction rules. For example, in a Vickrey auction, agents have dominant strategies while in a first-price-sealed-bid auction, an agent's optimal strategy depends on what strategies other agents are following.

3. The GVA announces the winners and their payment  $p_i$ :

$$p_i = \sum_{j \neq i} v_j(y'_j) - \sum_{j \neq i} v_j(y_j^*)$$

where  $Y' = (y'_1, \dots, y'_{i-1}, y'_{i+1}, \dots, y'_{|I|})$  is the allocation that maximizes the sum of all agents' valuations assuming that agent  $i$  did not participate.

Under the usual assumption that each agent has quasilinear preferences  $u_i(y_i) = v_i(y_i) - p_i$ , the utility of bidder  $i$  in the GVA is

$$u_i(y_i^*) = v_i(y_i^*) - p_i = v_i(y_i^*) + \sum_{j \neq i} v_j(y_j^*) - \sum_{j \neq i} v_j(y'_j).$$

The GVA has several nice properties for rational agents. First, if the agents have quasilinear preferences, the GVA is incentive compatible. The dominant strategy for rational agents is to bid their true valuations for the bundles of items. Second, the GVA is Pareto efficient. There is no other way to allocate the items (and compute payments) that would make some agent better off without making some other agent worse off. Finally, it is individually rational for agents to participate. An agent's utility obtained from participating in the GVA is never lower than if it had not participated (that is, the agent will never end up paying more for its bundle of items than its true valuation for the bundle).

### 2.2.3 COMPARISON OF AUCTIONS

The auctions described in the previous sections have been well studied for rational agents and the equilibria have been determined. The ascending auction, Vickrey auction and the generalized Vickrey auction all have dominant strategy equilibria or optimal best response equilibria [42, 17]. This means that agents participating in the auction do not have to guess what strategies other agents are following in order to play optimally. The first-price-sealed-bid auction has only Nash equilibria. An agent's optimal strategy depends on the strategies of the other agents. Table 1 shows which auctions have dominant strategy equilibria for rational agents.

## 3. Value Discovery

To bid sensibly, agents must have a value for the items being auctioned. How are these values obtained? In this paper we focus on settings where agents do not simply know (or are told) their valuations. Instead agents must actively allocate resources to determine them [38, 39]. If agents know their values *a priori* or are able to determine them with ease, then they can simply follow the equilibrium strategies for fully rational agents. However, in many situations agents have restrictions placed on their capabilities which affect how they are able to determine the values for the items. In the rest of the paper we will assume that agents are computational in nature. That is, agents determine their valuations by computing.

### 3.1 Role of Computing

In our model agents compute to determine their value for items up for auction.<sup>5</sup> While agents will be mainly interested in computing to determine their own values, they are also able to use their resources to (partially) determine the values of competitors. However, no matter how an agent uses its resources, it is important to first distinguish how computing affects the values.

We assume that an outcome from computing is represented as a *feature vector*.

**Definition 3 (Feature Vector)** A feature vector for agent  $i$  at time  $t$ ,  $\bar{f}_i(t)$ , is

$$\bar{f}_i(t) = (\text{insts}(t), \text{sol}(t)) \in \mathbb{I} \times \mathbb{S}$$

where  $\mathbb{I}$  is the set of features describing problem instances and  $\mathbb{S}$  is the set of features describing problem solutions. We define  $\mathbb{F}_i(t)$  to be the set of feature vectors at time  $t$ .

To illustrate the difference between problem instance features and solution features, consider the traveling salesman domain. The domain of problem instance features includes the average distance between cities and the variance in these distances while a solution feature is the length of a computed tour. Given a feature vector, we assume that agents are able to extract information from it in order to have a value for the item for auction. We denote the value of agent  $i$ , given feature vector  $\bar{f}_i(t)$ , by  $v_i(\bar{f}_i(t))$ . This value function is domain dependent. For example, in a traveling salesman problem, the value of a feature vector may be the length of the present tour as that determines how far the salesman has to drive. While we put few restrictions on how agents use the feature vectors to determine their values, we do assume that values are derived entirely from information in the feature vectors.

Computing can be used to *improve* or *refine* values. When agents compute to improve values, they must have some initial value for the item. We usually assume that this value is 0, but it need not be. As the agent computes the value of the item improves.

**Definition 4 (Improving)** Computing improves an agent's value if for all amounts of time  $t, t'$  such that  $t' \geq t$

$$v(\bar{f}(t')) \geq v(\bar{f}(t)).$$

Computing can also be used to *refine* values. When agents refine their values they start with an initial probability distribution over the interval where the true value lies. By computing, the agent refines the probability distribution until it learns the true value of the item by either obtaining the item and determining the true value, or refining the distribution of the value to a single point mass.

**Definition 5 (Refining)** Computing refines an agent's value if, by computing, the variance of the distribution from which the agent learns the value is drawn, is reduced. Formally, let  $F^t, F^{t'}$  be the probability distributions over an agent's value at time  $t$  and  $t'$ , and let  $\sigma_t^2, \sigma_{t'}^2$  be the variance of the respective distributions. Then, if  $t' \geq t$

$$\sigma_{t'}^2 \leq \sigma_t^2.$$

There is a subtle difference between computing to improve values and computing to refine values. When an agent computes to improve a value, if it is allocated the item, then the item's value is equal to the computed value. For example, in a traveling salesman domain, an agent's value may be equal to the inverse of the distance of the tour. As an agent computes, it finds shorter tours. When the agent is chosen to do the job, then its value is determined by the tour that it has computed. If the agent refines its value, the item has a value and the agent is computing to determine it. If the agent is allocated the item then it learns the true value. For example, in the art world the value of a painting depends, among other things, on whether it is a forgery. A bidder may have an initial guess as to whether the painting in question is a forgery or not, but by researching (computing) the bidder is able to better ascertain the likelihood of whether the painting is forged. This computing does not change the value of the painting, but it allows the bidder to submit a more informed bid.

---

5. Another interpretation is that agents are gathering information about their values. Everything discussed in this section also applies in the information acquisition model.

### 3.2 Restrictions on Computing

Agents often have restrictions on their available resources which effect how they can use them. In this paper we study a general restriction on agents' computing power. We investigate a setting where agents have a cost associated with computing which means that they may not be able to optimally solve their value problems because the expense becomes too high.

Assume that there are  $n$  possible problems on which an agent can compute. Define  $T^n$  to be the set of vectors  $(t_1, \dots, t_n)$  such that  $t_i$  is the amount of time spent computing of problem  $i$ .

**Definition 6 (Costly Computing)** *An agent  $i$  has costly computing if there exists a cost function,  $\text{cost}_i$ ,*

$$\text{cost}_i : T^n \mapsto \mathbb{R}^+ \cup \{0\}.$$

In general there are no restrictions placed on the cost functions of agents. However, there exists a family of cost functions that are of special interest. Let  $D$  be some amount of time. For any  $\bar{t} \in T^n$  define cost function,  $\text{cost}^D$  to be

$$\text{cost}^D(\bar{t}) = \begin{cases} 0 & \text{if } \sum_{j=1}^n t_j \leq D, \\ \infty & \text{otherwise.} \end{cases}$$

Such a cost function allows an agent to compute freely up to time  $D$ . After time  $D$  the agent must stop. This is exactly the situation where agents have hard deadlines. Since this is a situation that arises often in practice, we study it in detail.

**Definition 7 (Limited Computing)** *An agent has limited computing if at some time  $D$ , it must stop computing on all problems. That is, its cost function is  $\text{cost}^D$ .*

### 3.3 Controlling Computing

When agents have restricted computing capabilities there are tradeoffs that they must make in order to use the available resources in the best possible way. In particular, agents want to be able to find solutions that lead to decisions that “the least-cost, or best-return decision, net of computational cost” [41]. While the type of restrictions placed on the agents are exceedingly important to the agents, the tools that agents are supplied with in order to best use the resources are also important. In this section we describe different dimensions in the space of tools available to agents.

#### 3.3.1 ANYTIME ALGORITHMS

We assume that agents have algorithms that allow them to make a tradeoff between their computing resources and the quality of solutions (values). In particular we assume that agents have *anytime algorithms*. Anytime algorithms have the property that they return better quality solutions as more time is allocated to them. All anytime algorithms fall into one of two categories, interruptible algorithms or contract algorithms.

**Definition 8 (Interruptible Algorithms)** *Let  $\mathcal{A}$  be an anytime algorithm, and let  $v_{\mathcal{A}}(t)$  be the quality of the results obtained by stopping the algorithm after time  $t$ . Algorithm  $\mathcal{A}$  is an interruptible anytime algorithm if for all time  $t, t'$  if  $t' \geq t$  then*

$$v_{\mathcal{A}}(t') \geq v_{\mathcal{A}}(t).$$

**Definition 9 (Contract Algorithms)** *Let  $\mathcal{A}$  be an anytime algorithm and let  $v_{\mathcal{A}}(t)$  be the quality of results at time  $t$ . Algorithm  $\mathcal{A}$  is a contract algorithm if,*

1. *For any times  $t, t', t' \geq t$ , announced prior to the start of the algorithm*

$$v_{\mathcal{A}}(t') \geq v_{\mathcal{A}}(t),$$

2. *and, for any time  $t^* \neq t, t'$ , if algorithm  $\mathcal{A}$  is stopped at time  $t$  there is no guarantee on the quality of results.*

There is one substantial difference between interruptible and contract algorithms. Interruptible algorithms do not require the run-time to be determined in advance. They can be stopped at any time and are guaranteed to return a useful result. Contract algorithms, however, need to know in advance how much time has been allocated. A contract algorithm will optimize for the allocated amount of time, but makes no guarantees as to the solution quality if the algorithm is stopped at some other time point.

Many algorithms have anytime properties. Most iterative refinement algorithms are interruptible since they are always able to return a solution, and improve it if allowed to run longer [26]. There exist tree search algorithms that can be modeled as contract algorithms. For example, RTA\* uses a predetermined search horizon that is computed from a given time allocation [16]. While the algorithm can return a result for any time allocation, if it is interrupted before the time allocated then it may not return a result.

### 3.3.2 PERFORMANCE PROFILES

While anytime algorithms are models that allow for the trading off of computing for solution quality, they do not provide a complete solution for agents. Instead, anytime algorithms are paired with a meta-level control procedure that 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 time affects the output of the algorithm, and a process for using the information in the performance profile to make decisions. For the rest of the paper we will use the term performance profile to refer to both the descriptive and procedural aspects.

For an agent to make good decisions about how to use its computing resources, it is important that it know what results computing has currently given it. We assume that each agent keeps this information in a *state of computing*.

**Definition 10 (State of Computing)** *Assume agent  $i$  has  $n$  possible problems on which it can compute. The state of computing of agent  $i$  at time vector  $\bar{t} = (t_1, \dots, t_n)$  is*

$$\theta_i(\bar{t}) = \langle \bar{f}_1(t_1), \dots, \bar{f}_n(t_n) \rangle.$$

Let  $\Theta_i(t)$  be the set  $\Theta_i(t) = \{\theta_i(\bar{t}) \mid t = \sum_{i=1}^n t_i\}$ .

As mentioned earlier, the descriptive part of the performance profile describes how the results of an anytime algorithm change as more time is allocated. This helps an agent decide how much time it should spend computing on a particular problem. At the abstract level, a performance profile is a procedure that potentially uses information about current results to determine a probability distribution over future results.

**Definition 11 (Performance Profile)** *Let agent  $i$  be planning to compute on problem  $j$ . Let  $\mathbb{F}_i^j(t)$  be the set of feature vectors for problem  $j$  at time  $t$ ,  $T$  be the set of time steps and  $\Theta_i = \cup_{t \in T} \Theta_i(t)$ . The performance profile for agent  $i$  and problem  $j$  is*

$$PP_j^i : T \times \Theta_i \mapsto \Delta \mathbb{F}_i^j.$$

That is,  $PP_j^i(t, \theta_i(\bar{t})) = \Delta \mathbb{F}_i^j(t)$ .

Definition 11 is very abstract and does not state how the probability distribution is determined nor what information from the state of computing is used. There are numerous ways of doing so, and in the rest of this section we outline some commonly used performance profiles.

The *deterministic performance profile* is the simplest implementation of a performance profile. There is no uncertainty as to what feature vector will result for any time allocation. By abusing notation, a deterministic performance profile for agent  $i$  and problem  $j$ , can be represented as

$$PP_j^i(t) = \bar{f}_j(t).$$

In particular, knowledge of the state of computing plays no role in the outcome. Figure 1 illustrates a deterministic performance profile. With a deterministic performance profile, an agent can determine the optimal

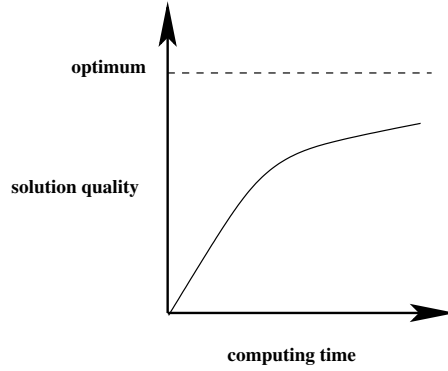


Figure 1: A deterministic performance profile for an anytime algorithm. For ease of illustration, the y axis shows solution quality, or  $v(\bar{f}(t))$  instead of feature vectors,  $\bar{f}(t)$ .

time allocation given its cost function, before doing any computing. The deterministic performance profile is useful for predicting solution quality only where there is no uncertainty in an algorithm's execution, and all instance features have been completely observed. Sadly, this is often not the case. Algorithms often incorporate randomization and often it is not possible to describe all features of a solution or instance due to lack of space or lack of knowledge. This can lead to some form of subjective uncertainty. The *stochastic performance profile* captures this uncertainty.

A stochastic performance profile maps allocation times to a probability distribution over feature vectors, that is

$$PP_j^i(t) = \Delta \mathbb{F}_i^j(t).$$

Like the deterministic performance profile, knowledge of an agent's state of computing plays no role in the outcome. An agent can determine the expected optimal time allocation given its cost function, before doing any computing. Figure 2 illustrates a stochastic performance profile.

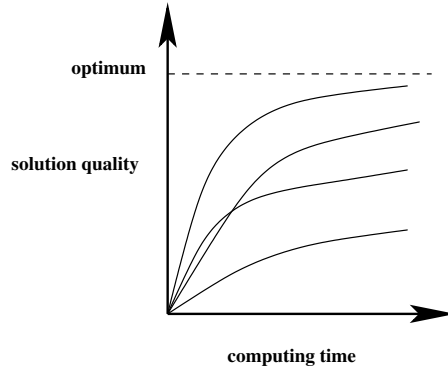


Figure 2: A stochastic performance profile for an anytime algorithm illustrating that there are multiple possible solutions for any allocation of time. The y axis shows solution quality instead of feature vectors. That is, the y axis is  $v(\bar{f}(t))$  instead of  $\bar{f}(t)$ .

The deterministic and stochastic performance profiles are ideally suited for contract algorithms as they allow agents to predict the future outcomes of algorithms. While they can also be used for interruptible

algorithms, often there is information in the states of computing of the agents that can be used to make better computing tradeoffs. Taking information from an algorithm’s progress into account when making decisions about allocating resources is called *conditioning*.

There are different levels of conditioning. An agent can condition using only the most recent results. Given a state of computing, the performance profile determines a probability distribution given the current feature vector for that problem. Formally, a performance profile,  $PP_j^i$  conditions on the most recent results if

$$PP_j^i(t_j + x, \theta_i(\bar{t})) = \Delta \mathbb{F}_i^j(t + x) | \bar{f}_j(t_j)$$

where  $\Delta \mathbb{F}_i^j(t + x) | \bar{f}_j(t_j)$  is the probability distribution over feature vectors at time  $t_j + x$  given that the present feature vector is  $\bar{f}_j(t_j)$ .

Performance profiles that condition on the most recent result are often represented as a table of discrete values which specifies a discrete probability distribution over solution vectors for each time step. While discretization is necessary in order to make any representation of a performance profile feasible, the table based approach loses some information that can be useful in conditioning. For example, information about the path of the algorithm is lost (Figure 3).

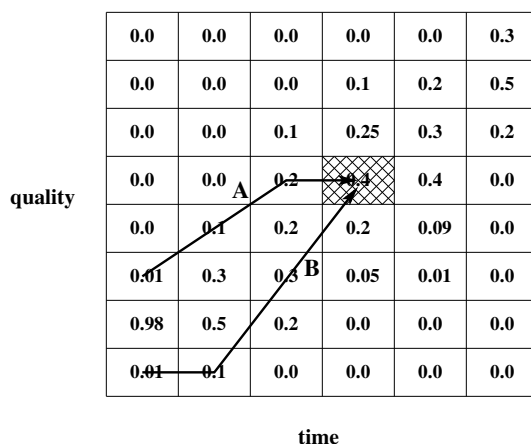


Figure 3: A table-based performance profile. While it is possible to condition on solution quality, information about the path is lost. For example, to reach the shaded square, an algorithm could have followed path A or path B. If it followed path B then it appears more likely that the solution quality would continue to improve at a faster rate than if path A had been followed.

There are a range of techniques for working with table-based performance profiles, ranging from a simple myopic approach that simply, given the current solution quality, makes its decision about computing based on one-step of lookahead [9], to more complex techniques involving dynamic programming [9].

To illustrate how such performance profiles work, we will describe in detail how the dynamic programming works. The goal is to obtain a monitoring policy  $\pi(q_i, t_k)$  which is a mapping from time step  $t_k$  and solution quality  $q_i$ . We define the utility of having a solution of quality  $q_i$  at time  $t_k$  to be  $U(q_i, t_k) = q_i - \text{cost}(t_k)$ . A stopping rule can be found by optimizing the following value function

$$V(q_i, t) = \max_d \begin{cases} U(q_i, t) & \text{if } d = \text{stop} \\ \sum_j Pr(q_j | q_i, \Delta t) V(q_j, t + \Delta t) & \text{if } d = \text{continue} \end{cases}$$

to determine the policy

$$\pi(q_i, t) = \arg \max_d \begin{cases} U(q_i, t) & \text{if } d = \text{stop} \\ \sum_j Pr(q_j | q_i, \Delta t) V(q_j, t + \Delta t) & \text{if } d = \text{continue} \end{cases}$$

Sometimes an agent may wish to condition their computing decisions on more information than just most recent solution quality. In particular the path of the solution may be of use.

**Definition 12 (Full conditioning)** A performance profile conditioned on path *uses all possible information in order to determine what the best computing action to take is. For example, it considers information about solution quality, other solution features, and the path of the algorithm.*

If agents wish to fully condition their computing actions on all information available (such as solution quality, problem instance, and path of an algorithm), they require a more general performance profile representation such as a *performance profile tree* [18, 20, 22].

A performance profile tree has two types of nodes, and edges that are both weighted and labeled. The first node type is the solution node. These nodes store the solution quality and feature vector returned by the algorithm at a certain point in time. The other node type is the random node. Random nodes represent the use of randomization which may influence the path of the algorithm. The edges emanating from a solution node always have weight equal to one. This represents the execution of one computing step. The edges from a random node are labeled with the probability of reaching a child, given that the parent was reached. The edges from a random node all have a weight of zero. This is because it is assumed that the randomization is not a computing step. The random node edges are each associated with some event, such as the drawing of a specific number. The labels on these edges are the probabilities with which the specific events occur. Figure 4 exemplifies one such tree.

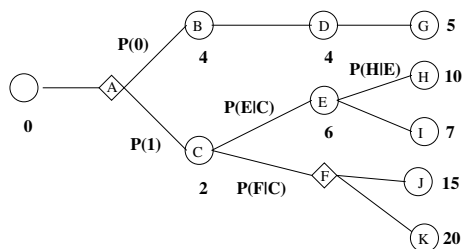


Figure 4: An agent's stochastic performance profile tree for a valuation problem. The diamond shaped nodes are random nodes and the round nodes are solution nodes. The feature vectors are not shown though the solution quality  $q(\vec{f})$  is shown for each solution node. At random node A, the probability 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.

Performance profile trees can support conditioning on any and all features that are deemed to be of importance by the agent. In particular any feature in the feature vector can be used for conditioning, as well as the solution quality of that feature vector. Another feature of the performance profile tree is that it supports conditioning on the path of solution so far. The performance profile tree that applies given a path of computing 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  $\{n' | n' \text{ is a node in } \mathcal{T}_i^g(n) \text{ which is reachable in } t \text{ time steps}\}$ .

## 4. Strategic Computing and Deliberation Equilibria

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 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 5 illustrates this relation.

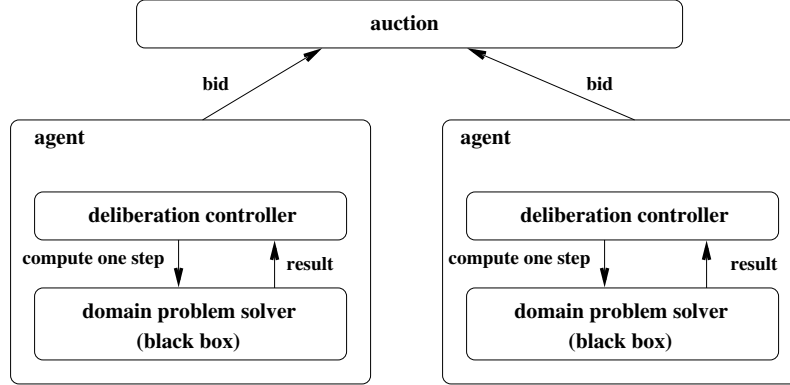


Figure 5: An auction with two computationally limited bidding agents. In order to submit a reasonable bid, each agent needs to first (approximately) compute its valuation for the item that is up for auction.

Let  $I$  be the set of agents and let  $G$  be the set of bundles of items that are being auctioned. In our model the game is divided into time periods. In each time period, an agent,  $i$ , is allowed to either execute a computing action,  $\text{comp}_i^{(g,j)}$ , or a computing action followed by a bidding action,  $(\text{comp}_i^{(g,j)}, b_i)$ . A computing action is the act of computing for one time step on some agent  $j$ 's valuation problem for bundle  $g$ . An agent may also decide to take a null computing action,  $\emptyset^C$ , by not computing on any problem. The bidding language and auction mechanism influence the type of bids submitted, but in general, an agent will submit a tuple which consists of either a numerical value for each bundle being auctioned, or a null value,  $\emptyset^B$ , indicating lack of interest in that particular bundle. That is, for agent  $i$  at time  $t$ ,  $b_i \in (\mathbb{R} \cup \emptyset)^{|G|}$ .

In general a history for agent  $i$  at time period  $t$ ,  $h_i(t)$ , is a list of all actions agent  $i$  has taken. We augment this definition of a history to include the cost incurred by the agent at time  $t$ ,  $\text{cost}_i(\bar{t})$ , and a state of computing at time  $t$ ,  $\theta_i(\bar{t})$ . We represent this augmented history by  $H_i(t) = (h_i(t), \text{cost}_i(\bar{t}), \theta_i(\bar{t}))$  and define  $\mathcal{H}_i(t) = \{H_i(t)\}$ .

A strategy is a mapping from history to action and the auction mechanism determines the legal strategies available to an agent. A sequential auction mechanism allows agents to interleave bidding and computing actions, while in a single-shot auction, the only time bidding is allowed is at the close of the auction.

**Definition 13** A strategy for agent  $i$  in an ascending auction is

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

where

$$\sigma_i(t+1) : \mathcal{H}_i(t) \times p(t) \mapsto (\{c^{(g,j)} \mid g \in G, j \in I\} \cup \{\emptyset^C\}) \times (\text{stay}, \text{fold})^{|G|}$$

where  $p(t)$  is the price at time  $t$ .

In a single-shot auction the strategy is different for two reasons. First, agents are only allowed to place one bid at the close of the auction. Secondly, agents have no opportunity to view other agents' bids.

**Definition 14** A strategy for agent  $i$  in a single-shot auction which closes at time  $T$  is

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

where

$$\sigma_i(t+1) : \begin{cases} \mathcal{H}_i(t) \times B_i(t) \mapsto (\{c^{(g,j)} | g \in G, j \in I\} \cup \{\emptyset^C\}) \times \{\emptyset^B\} & \text{if } t+1 \neq T \\ \mathcal{H}_i(t) \times B_i(t) \mapsto (\{c^{(g,j)} | g \in G, j \in I\} \cup \{\emptyset^C\}) \times (\mathbb{R} \cup \emptyset^B)^{|G|} & \text{otherwise} \end{cases}$$

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

Agents use their computing 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 computing*. The second we call *strong strategic computing*.

**Definition 15 (Strong Strategic Computing)** 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 computing. That is, a strategy,  $S_i = (\sigma_i(t))_{t=0}^{\infty}$ , consists of strong strategic computing if there exists a  $t$ , a history  $x \in \mathcal{H}_i(t) \times B_i(t)$  such that

$$\sigma_i(t)(x) = (c^{(g,j)}, b)$$

where  $j \neq i$ .

In strong strategic computing the agent uses its own computing 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. Weak strategic computing is different because agents do not actively compute on each others problems.

**Definition 16 (Weak Strategic Computing)** Assume there are two agents,  $i$  and  $j$ . Assume that agent  $j$  has two possible performance profiles,  $P_1$  and  $P_2$ . Let  $\mathcal{S}_i^{P_1}$  be the set of strategies for agent  $i$  if agent  $j$ 's performance profile is  $P_1$  and let  $\mathcal{S}_i^{P_2}$  be the set of strategies for agent  $i$  if agent  $j$ 's performance profile is  $P_2$ . Agent  $i$  is performing weak strategic computing if it 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. That is, agent  $i$  is not performing strong strategic computing and

$$\mathcal{S}_i^{P_1} \neq \mathcal{S}_i^{P_2}.$$

In weak strategic computing, an agent does not use its computing 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. An alternative formulation of strategic computing is to think of it as additional conditioning. Agents are provided with tools (performance profiles) that allow them to condition their computing on the results they obtain from computing on their own problems. Strategic computing allows them to further condition their strategies based on results they think their competitors may have obtained. Ideally, neither form of strategic computing is present in an auction. However, strong strategic computing 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 computing time (and therefore worse solutions) on their own actual problems. In economic terms, strong strategic computing generally decreases Pareto efficiency.

## 5. Results

In this section we study different auction types in order to understand how the restrictions placed on the bidding agents computing capabilities affect their strategies. There are some standard game-theoretic assumptions that are made in this section. Unless otherwise noted, we assume that the agents' performance profiles and cost functions are common knowledge. We also assume that given a feature vector for a problem of some agent, it is possible for all other agents to determine the value of that solution from the feature vector. We assume that all agents are risk neutral, that we are in a private value setting, and an agent's utility is defined as

$$u = \begin{cases} v - p - \text{cost} & \text{if the agent is allocated the item} \\ -\text{cost} & \text{otherwise} \end{cases}$$

where  $v$  is the (computed) value of the item,  $p$  is the price paid for the item, and  $\text{cost}$  is the cost the agent incurred by determining the value.

### 5.1 Ascending and Vickrey Auctions

If agents are fully rational, then in both the ascending and Vickrey auctions agents have dominant bidding strategies. In particular, the two auctions are strategically equivalent as it is possible to find an isomorphism between the equilibrium strategies. For computationally restricted agents the optimal bidding strategies for agents in both the ascending and Vickrey auction are similar to those of fully rational agents. Using arguments identical to those used in the fully rational setting, it is possible to show that in a Vickrey auction, a computationally restricted agent is best off submitting a bid equal to its computed value or equal to its expected computed value (since we assume agents are risk neutral). Similarly, an agent in an ascending auction is best off staying in the auction only until the price becomes greater than its computed value (or expected computed value). However, there is a potential difference between the computing strategies of agents in the auctions.

Consider the following simple example. Assume that there are two agents,  $\alpha$  and  $\beta$ . Each agent has their own cost function,  $\text{cost}_\alpha$  and  $\text{cost}_\beta$ , and performance profiles  $PP_\alpha$  and  $PP_\beta$ . Finally, assume that the performance profiles are such that after one step of computing each agent will know its true value.  $v_i \in [a_i, b_i]$ . In a Vickrey auction, an agent has two time points where it is useful to acquire information about its value. It can either acquire information before it places a bid, or it can bid its expected value and then wait to determine its true value only after it has been allocated the item.<sup>6</sup> Since the auction is sealed-bid, an agent learns nothing about its competitor's actions. In an ascending auction, the agents have the luxury of being able to wait for further information before deciding whether to compute on their value problem. In the small example, even without computing agent  $i$  knows that its value must be at least  $a_i$ . It need not do any computing to learn its value while the price is below  $a_i$  since, if it is allocated the item, its utility will surely be greater than zero. Therefore it is possible that there exist situations where agents would compute to find their valuations in a Vickrey auction, but would not need to in an ascending auction.

The previous discussion was independent of the type of restriction on the agents computing resources. We will now study whether having costly or limited computing affects agents equilibrium strategies.

#### 5.1.1 LIMITED COMPUTING

If agents have limited computing, then in both an ascending auction and a Vickrey auction no form of strategic computing occurs. Agents have (weakly) dominant strategies which have them computing only on their own value problems, independent of their own and other agents' performance profiles and deadlines, and whether they compute to improve or refine their values. We show this by first proving two lemmas and then combining the lemmas into the final theorem.

---

6. Recall, that if the agent computes to improve its value, then it *must* compute once it has acquired the item as the item has no value without the occurrence of computing. If the agent computes to refine its value, then once it is allocated the item, it need not compute. It learns the value of the item upon receipt.

**Lemma 1** *Assume agents have limited computing and compute to improve their values. In an ascending auction or an a Vickrey auction agents have (weakly) dominant strategies which involve no weak nor strong strategic computing.*

**Proof:** Since agents have limited computing, for each agent  $i$  there exists a deadline  $D_i$  such that the cost function of agent  $i$  is

$$\text{cost}_i(\bar{t}) = \begin{cases} 0 & \text{if } \sum_j t_j \leq D_i \\ \infty & \text{if } \sum_j t_j > D_i \end{cases}$$

This means that agent  $i$  can compute up to  $D_i$  time steps for free, but will never compute more than  $D_i$  steps as then  $u_i = -\infty$ . Assume that agent  $i$  computes  $k_i \leq D_i$  on its own problem. Then, its value is  $v_i(\bar{f}_i(k))$ .

First, consider a Vickrey auction. Assume that the highest value among the set of agents not including agent  $i$  is  $b$ . If  $v(\bar{f}_i(k)) > b$  then agent  $i$  is allocated the object and  $u_i = v(\bar{f}_i(k)) - b$ . Since agents are computing to improve their values,  $v_i(\bar{f}_i(k)) \leq v_i(\bar{f}_i(D_i))$  and so agent  $i$  would have higher utility if it computed  $D_i$  steps on its own problem. If  $b > v_i(\bar{f}_i(k))$  then  $u_i = 0$ . If agent  $i$  computed for  $D_i$  steps then either  $b > v(\bar{f}_i(D_i))$  and so  $u_i = 0$  or  $b \geq v(\bar{f}_i(D_i))$  and  $u_i \geq 0$ . Computing on another agent's problem does not change the second highest bid, and computing less time on its own problem only lowers its utility, therefore in a Vickrey auction agents should compute only on their own problem until their deadline.

In an ascending auction the argument is identical. □

**Lemma 2** *Assume agents have limited computing and compute to refine their values. In an ascending auction or Vickrey auction agents have (weakly) dominant strategies which involve no weak nor strong strategic computing.*

**Proof:** Since agents have limited computing, for each agent  $i$  there exists a deadline  $D_i$  such that the cost function of agent  $i$  is

$$\text{cost}_i(\bar{t}) = \begin{cases} 0 & \text{if } \sum_j t_j \leq D_i \\ \infty & \text{if } \sum_j t_j > D_i \end{cases}$$

This means that agent  $i$  can compute up to  $D_i$  time steps for free, Agent  $i$  will not compute for more than  $D_i$  steps since then  $u_i = -\infty$ .

Since agents are computing to refine their values, once an agent has been allocated an item it learns the true value. Agents have no incentive to compute after the item has been allocated since either they know the value, or they were not allocated the item. This means that in a Vickrey auction agents compute before submitting a bid, and in an ascending auction agents compute sometime before withdrawing from the auction.

Assume agent,  $i$  has computed  $k_i \leq D_i$  steps on its own problem. Let  $v_i$  denote the true value of the item if it is allocated to agent  $i$  and let  $E[v_i|k_i]$  denote the expected value of the item to agent  $i$  after it has computed  $k_i$  steps. Let  $b$  be the maximum bid submitted by the other agents in the Vickrey auction. An agent is always best off by bidding its expected value. The optimal situation is when the bid submitted is equal to the true value of the item, that is when  $E[v_i|k_i] = v_i$ . In such a case, the agent's utility is

$$u_i = \Pr(v_i \geq b)(v_i - b).$$

If  $E[v_i|k_i] < v_i$  then the utility for agent  $i$  is

$$u_i = \Pr(E[v_i|k_i] \geq b)(v_i - b) \leq \Pr(v_i \geq b)(v_i - b).$$

If  $E[v_i|k_i] > v_i$  then the utility for agent  $i$  is

$$\begin{aligned} u_i = \Pr(E[v_i|k_i] \geq b)(v_i - b) &= \Pr(v_i \geq b)(v_i - b) - \Pr(E[v_i|k_i] \geq b > v_i)(b - v_i) \\ &\leq \Pr(v_i \geq b)(v_i - b). \end{aligned}$$

For the agent to maximize its utility, it must compute so as to have the computed expected value as close to the real value of the item as possible. From the Chebyshev Inequality, it is known that for any  $y \geq 0$ , the probability that a random variable  $v_i$  will be at most distance  $y$  away from  $E[v_i]$  is

$$Pr(|v_i - E[v_i]| > y) \leq \frac{\sigma^2}{y^2}$$

where  $\sigma^2$  is the variance.

Because agents are computing to refine their values, for all  $t \leq D_i$  and for all  $y \geq 0$

$$\frac{\sigma^2(D_i)}{y^2} \leq \frac{\sigma^2(t)}{y^2}.$$

Thus, if an agent wishes to maximize its utility it is best off computing for  $D_i$  time steps on its own problem.

In an ascending auction the argument is identical. □

Combining Lemmas 1 and 2 we derive the following theorem.

**Theorem 1** *If agents have limited computing then in both the ascending and Vickrey auctions there always exist (weakly) dominant strategy equilibria where neither strong nor weak computing occurs.*

### 5.1.2 COSTLY COMPUTING

If agents incur a cost while computing then their optimal strategies may be quite different. It may be of use for an agent to actually compute on other agents' problems in order to learn what their values are. That is, strategic computing may occur in equilibrium.

To illustrate the phenomena we present a simple scenario. Assume that there are two agents,  $\alpha$  and  $\beta$ , participating in an ascending auction (The argument for a Vickrey auction is identical except that the timing of the computing may be different.) Assume that agent  $\alpha$  has the performance profile  $PP_\alpha$  such that before computing it knows that its value,  $v_\alpha$ , after one step lies in the interval  $[a, b]$ , and the probability that its value is  $v_\alpha = x$  is  $f(x)$ . After one step of computing it knows its true value. Assume that agent  $\beta$  has a performance profile  $PP_\beta$  such that before computing it knows that its value,  $v_\beta$ , after one step lies in the interval  $[c, d]$  where  $c \leq a$ , and the probability that its value of  $v_\beta = y$  is  $g(y)$ . After one step of computing it knows its true value. Finally, assume that the cost function of agent  $\alpha$  is

$$\text{cost}_\alpha(\langle t_\alpha, t_\beta \rangle) = Bt_\beta \quad \text{where } B > b$$

and the cost function of agent  $\beta$  is

$$\text{cost}_\beta(\langle t_\alpha, t_\beta \rangle) = t_\alpha + 2t_\beta.$$

It is easy to show that agent  $\alpha$  has a dominant strategy which is to compute one step on its own problem. Since the cost of computing on its competitor's,  $\beta$ , problem is higher than its highest possible computed value, agent  $\alpha$  will never compute on it. If agent  $\alpha$  chooses to not participate then its utility is

$$u_\alpha^{\text{nothing}} = 0.$$

In expectation, for agent  $\alpha$  the strategies of bidding blindly, or computing one step on its problem to determine its value have the same utility. That is, if agent  $\beta$  submits a bid,  $b^*$ , then

$$u_\alpha^{\text{blind}} = u_\alpha^\alpha = \int_a^b \int_c^{v_\alpha} (v_\alpha - b^*) f(v_\alpha) g(b^*) db^* dv_\alpha \geq 0.$$

Therefore, agent  $\alpha$  is best off computing on its own problem only and staying in the auction until the price reaches its computed value.

To determine the best-response for agent  $\beta$ , one must recall that each computing step it takes incurs a cost. While in an ascending auction, agent  $\beta$  can wait until the price reached  $c$  before making a decision as to whether to compute or not, since in this example  $c \leq a$ , waiting provides no information about agent  $\alpha$ 's value.

Agent  $\beta$  always has the option of doing nothing. In such a situation, its utility is

$$u_{\beta}^{\text{nothing}} = 0. \quad (1)$$

Agent  $\beta$  may decide to compute only on its own problem. If it does decide to do this, then it will compute only one step since more steps incur a cost but do not refine or improve its value. Following this strategy results in expected utility

$$u_{\beta}^{\beta} = -2 + \int_c^d \int_a^{v_{\beta}} (v_{\beta} - v_{\alpha}) f(v_{\alpha}) dv_{\alpha} g(v_{\beta}) dv_{\beta}. \quad (2)$$

Agent  $\beta$  may decide to bid blindly. By this, we mean that agent  $\beta$  may decide to not compute on its value before bidding. Instead, it may bid only using its expected value. If it is allocated the item then under the improving model it must compute on its own problem, while under the refining model it does not need to compute since it learns its value once it obtains the item. Let

$$V = \int_c^d v_{\beta} g(v_{\beta}) dv_{\beta}.$$

The utility for agent  $\beta$  under the improving model is

$$u_{\beta}^{\text{blind\_improve}} = \int_c^d \int_a^V (v_{\beta} - v_{\alpha} - 2) f(v_{\alpha}) dv_{\alpha} g(v_{\beta}) dv_{\beta}. \quad (3)$$

and the utility for agent  $\beta$  under the refining model is

$$u_{\beta}^{\text{blind\_refine}} = \int_c^d \int_a^V (v_{\beta} - v_{\alpha}) f(v_{\alpha}) dv_{\alpha} g(v_{\beta}) dv_{\beta}. \quad (4)$$

Finally, it is possible that agent  $\beta$  will compute on agent  $\alpha$ 's problem. It will never compute on agent  $\alpha$ 's problem after it knows its own value since once it knows its own value it can bid optimally. However, there may exist situations where it can use information about agent  $\alpha$ 's problem to help decide whether to compute on its own problem. In particular, if agent  $\beta$  learns that the value for agent  $\alpha$  is greater than some threshold  $v^*$  then agent  $\beta$  may decide not to compute on its own problem since the likelihood of it winning the auction is small, and so it wishes to avoid the cost of computing. The utility for agent  $\beta$  following a strategy where it first computes on agent  $\alpha$ 's value problem, and then computes on its own problem only if  $v_{\alpha} \leq v^*$  is

$$u_{\beta}^{\alpha-\beta} = - \int_{v^*}^b f(v_{\alpha}) dv_{\alpha} - 3 \int_c^{v^*} \int_a^{v^*} f(v_{\alpha}) dv_{\alpha} g(v_{\beta}) dv_{\beta} + \int_{v^*}^d \int_a^{v_{\beta}} (v_{\beta} - v_{\alpha}) f(v_{\alpha}) dv_{\alpha} g(v_{\beta}) dv_{\beta} \quad (5)$$

where

$$v^* = \arg \max_x \left[ - \int_x^b f(v_{\alpha}) dv_{\alpha} - 3 \int_c^x \int_a^x f(x) dx g(v_{\beta}) dv_{\beta} + \int_x^d \int_a^{v_{\beta}} (v_{\beta} - v_{\alpha}) f(v_{\alpha}) dv_{\alpha} g(v_{\beta}) dv_{\beta} \right].$$

Agent  $\beta$  will partake in strong strategic computing only when Equation 5 is greater than Equations 1, 2, and 3 or 4. The question is ‘‘Does there exist actual performance profiles such that strong strategic computing occurs?’’

Using the performance profiles that were mentioned earlier, let agent  $\alpha$ 's performance profile be  $[a, b] = [12, 30]$  with the distribution

$$f(x) = \begin{cases} p & \text{if } x = 30 \\ 1 - p & \text{if } x = 12 \\ 0 & \text{otherwise.} \end{cases}$$

Let the performance profile of agent  $\beta$  be  $[3, 22]$  with distribution

$$g(x) = \begin{cases} q & \text{if } x = 22 \\ 1 - q & \text{if } x = 3 \\ 0 & \text{otherwise.} \end{cases}$$

Using these performance profiles, it is possible to determine the utility agent  $\beta$  would have from following the strategies outlines above. That is

$$u_{\beta}^{\text{nothing}} = 0$$

$$u_{\beta}^{\beta} = -2 + 10(1 - p)q$$

$$u_{\beta}^{\text{blind\_imp}} = 8(1 - p)q - 11(1 - p)(1 - q)$$

or

$$u_{\beta}^{\text{blind\_ref}} = 10(1 - p)q - 9(1 - p)(1 - q)$$

and

$$u_{\beta}^{\alpha-\beta} = -p + 7(1 - p)(1 - q) - 3(1 - p)(1 - q).$$

It is possible to determine for which values of  $p$  and  $q$  each strategy is dominant. In the case where agents compute to improve their values, each strategy is the dominant strategy for agent  $\beta$  is the following regions.

**1. Nothing**

$$0 < p \leq \frac{1}{2}, \quad 0 < q < \frac{-1}{5p - 5}$$

$$\frac{1}{2} < p \leq \frac{53}{72}, \quad 0 < q < \frac{2p - 3}{10p - 10}$$

$$\frac{53}{72} < p < 1, \quad 0 < q < \frac{11}{19}$$

**2. Blind**

$$0 < p \leq \frac{1}{2}, \quad \frac{11p - 9}{9p - 9} < q < 1$$

$$\frac{1}{2} < p \leq \frac{53}{72}, \quad \frac{9p - 8}{9p - 9} < q < 1$$

$$\frac{53}{72} < p < 1, \quad \frac{11}{19} < q < 1$$

**3. Alpha Beta**

$$\frac{1}{2} < p < \frac{53}{72}, \quad \frac{2p - 3}{10p - 10} < q < \frac{9p - 8}{9p - 9}$$

**4. Beta**

$$0 < p < \frac{1}{2}, \quad \frac{-1}{5p - 5} < q < \frac{11p - 9}{9p - 9}$$

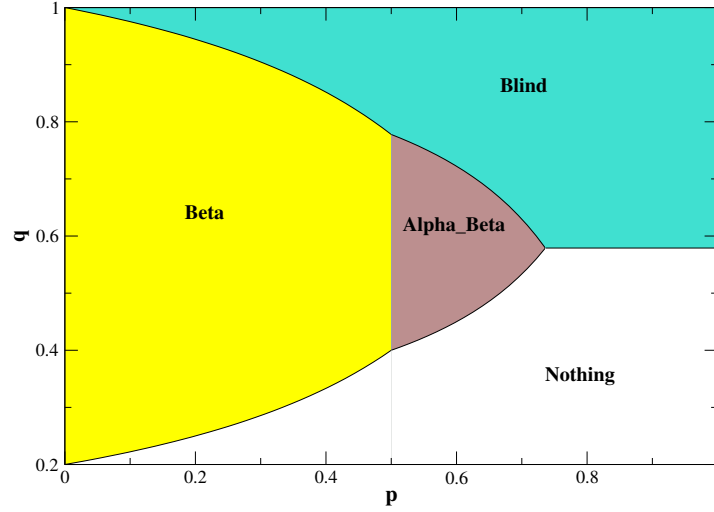


Figure 6: Agents are computing to improve their valuations. For different values of  $p$  and  $q$ , different strategies for agent  $\beta$  are the best response to agent  $\alpha$ 's dominant strategy. If  $\frac{1}{2} < p < \frac{53}{72}$  and  $\frac{2p-3}{10p-10} < q < \frac{9p-8}{9p-9}$  agent  $\beta$  is best off computing one step on agent  $\alpha$ 's problem.

The regions where different strategies are dominant are shown in Figure 6.

If the two agents are computing to *refine* their values, then each of agent  $\beta$ 's strategies are dominant in the following regions.

1. **Nothing**

$$0 < p \leq \frac{1}{2}, \quad 0 < q < \frac{-1}{5p-5}$$

$$\frac{1}{2} < p \leq \frac{33}{52}, \quad 0 < q < \frac{2p-3}{10p-10}$$

$$\frac{33}{52} < p < 1, \quad 0 < q < \frac{9}{19}$$

2. **Beta**

$$0 < p < \frac{1}{2}, \quad \frac{-1}{5p-5} < q < \frac{9p-7}{9p-9}$$

3. **Alpha\_Beta**

$$\frac{1}{2} < p < \frac{33}{52}, \quad \frac{2p-3}{10p-10} < q < \frac{7p-6}{9p-9}$$

4. **Blind**

$$0 < p \leq \frac{1}{2}, \quad \frac{9p-7}{9p-9} < q < 1$$

$$\frac{1}{2} < p \leq \frac{33}{52}, \quad \frac{7p-6}{9p-9} < q < 1$$

$$\frac{33}{52} < p < 1, \quad \frac{9}{19} < q < 1$$

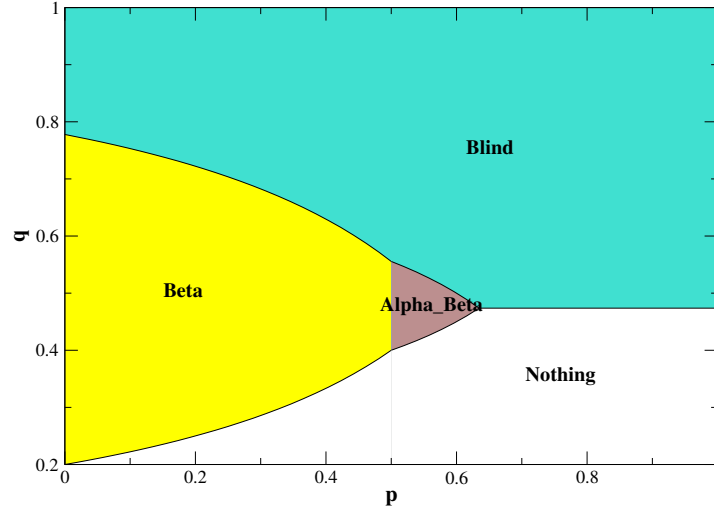


Figure 7: Agents are computing to refine their values. For different values of  $p$  and  $q$ , different strategies for agent  $\beta$  are the best response to agent  $\alpha$ 's dominant strategy. If  $\frac{1}{2} < p < \frac{33}{52}$  and  $\frac{2p-3}{10p-10} < q < \frac{7p-6}{9p-9}$  agent  $\beta$  is best off computing one step on agent  $\alpha$ 's problem.

Figure 7 illustrates these regions.

The example that has just been presented made no assumption as to the type of performance profile used, other than to assume that it was not deterministic. The example did not make any assumptions on whether the anytime algorithm was a contract or interruptible algorithm either. The following theorem has been proved.

**Theorem 2** *If agents have anytime algorithms (contract or interruptible), costly computing and their performance profiles are not deterministic then in ascending and Vickrey auctions strong strategic computing may occur in equilibrium. This result holds whether agents are computing to improve or refine their values.*

Theorem 2 only is true if the performance profiles are not deterministic. Agents compute on each others' problems in order to remove uncertainty as to their competitors possible values. If all agents have deterministic performance profiles then all agents know what the results of computing are with certainty.

**Lemma 3** *If agents have anytime algorithms (contract or interruptible), costly computing, and their performance profiles are deterministic, then in an ascending or Vickrey auction no strong strategic computing occurs. This result holds whether agents compute to improve or refine their values.*

**Proof:** If agents compute to refine their values, then, if their performance profiles are deterministic agents do not need to compute. Their performance profiles tell the agents with certainty what their values are, without having to compute and thus incur a cost. Trivially, since agents do not compute, there is no strong strategic computing.

Assume agents compute to improve their values. Since all performance profiles and cost functions are common knowledge each agent is able to determine what each agent's value function and cost function is. If agent  $j$  computes on agent  $i$ 's value problem, it does not obtain any new information, but does incur a cost, thus reducing its utility. Therefore, no strong strategic computing occurs.  $\square$

Even though there is no strong strategic computing, weak strategic computing can still be a problem.

**Lemma 4** *If agents have anytime algorithms (contract or interruptible), costly computing, and their performance profiles are deterministic, then in an ascending or Vickrey auction weak strategic computing can occur in equilibrium. This result holds whether agents compute to improve or refine their values.*

**Proof:** Assume there are two agents,  $\alpha$  and  $\beta$ , competing in either an ascending or Vickrey auction, with deterministic performance profiles  $PP_\alpha$  and  $PP_\beta$ . Assume that the cost functions of the agents are common knowledge. Using the deterministic performance profiles, each agent is able to determine

$$t_\alpha^* = \arg \max_{t_\alpha} [v_\alpha(\bar{f}_\alpha(t_\alpha)) - \text{cost}_\alpha(\langle 0, \dots, 0, t_\alpha, 0, \dots, 0 \rangle)]$$

and

$$t_\beta^* = \arg \max_{t_\beta} [v_\beta(\bar{f}_\beta(t_\beta)) - \text{cost}_\beta(\langle 0, \dots, 0, t_\beta, 0, \dots, 0 \rangle)]$$

In a Vickrey auction, agent  $i$  will submit a bid equal to  $v_i(\bar{f}_i(t_i))$ , and in an ascending auction an agent will drop out when the price reaches  $v_i(\bar{f}_i(t_i))$ . If agent  $\beta$  knows (from  $PP_\alpha$ ) that

$$v_\alpha(\bar{f}_\alpha(t_\alpha)) > v_\beta(\bar{f}_\beta(t_\beta))$$

then it will not compute because it is certain to lose to auction and thus have utility

$$u_\beta = -\text{cost}_\beta(\langle 0, \dots, 0, t_\beta, 0, \dots, 0 \rangle)$$

Since agent  $\beta$ 's utility from not computing is 0, its best strategy is to not participate. That is, agent  $\beta$  determines its strategy using information from agent  $\alpha$ 's performance profile. In other words, weak strategic computing can occur.  $\square$

Combining Lemmas 3 and 4 leads to Theorem 3

**Theorem 3** *If agents have anytime algorithms (contract or interruptible), costly computing, and their performance profiles are deterministic, then in an ascending or Vickrey auction weak strategic computing can occur in equilibrium but strong strategic computing does not. This result holds whether agents compute to improve or refine their values.*

## 5.2 First-price Sealed-bid Auctions

In a first-price sealed-bid auction each agent submits one bid without knowing what the other agents' have bid. The highest bidder wins the item and pays the amount of its bid. For fully rational agents there is no dominant bidding strategy for agents. An optimal strategy depends on the bids of the other agents. If an agent knows what the submitted bids of other agents are, then it can tailor its bid so as to maximize its own utility. Not surprisingly, computing agents may have incentive to use some of their computing resources in order to discover what valuations the other agents may have for the item up for auction as allowing agents to compute on each others problems simply supplies the agents with a strong tool for concrete speculation. This is independent of whether the agents have costly or free but limited computing resources, whether agents must do all their computing before the auction occurs or whether they are allowed to compute after the bids have been submitted and a winner has been determined. The agents deliberation and bidding strategies depend upon the performance profiles that the agents have for their valuation problems.

While in this auction strong strategic computing is the norm, situations where there is no strong strategic computing is of interest. We find that in general, for the first-price sealed-bid auction the only time when agents do not use strong strategic computing is when the performance profiles of any type contain no uncertainty. That is, that all the performance profiles can be transformed to deterministic performance profiles. This is independent of whether computing is free or limited, and whether agents compute to improve or refine their valuations.

**Theorem 4** *Assume that agents in a first-price sealed-bid auction all have deterministic performance profiles. Then only weak strategic computing will occur in Nash equilibrium. This is independent of whether agents have costly or limited computing and whether agents are computing to improve or refine their valuations.*

**Proof:**

Assume that all agents are computing to refine their valuations. Assume also that all agents have deterministic performance profiles. As there is no uncertainty to be removed by computing, no strong strategic computing will occur either. However, weak strategic computing can occur as agents' can check each others' performance profiles so as to learn what the different valuations are.

Assume that agents are computing to improve their valuations. If the agents all have deterministic performance profiles then all agents can determine what valuations the other agents will be able to compute, given their deadlines (weak strategic computing). Agents have (weakly) dominant deliberation strategies where they compute solely on their own problem. Overall, the agents strategies will not be (weakly) dominant as the bid submitted by one agent depends on what it believes the other agents will submit as bids.  $\square$

**5.3 Generalized Vickrey Auction**

The generalized Vickrey auction (GVA) inherits some of the properties of the single item Vickrey auction. For rational agents, the dominant bidding strategy is to submit bids which are equal to the agents' true values. For agents with bounded resources for determining values, properties from the single item Vickrey auction also are prevalent in the GVA. In particular, if agents have costly computing resources then strong strategic computing may occur in equilibrium.

**Theorem 5** *If agents have anytime algorithms, costly computing and their performance profiles are not deterministic, then in the generalized Vickrey auction strong strategic computing can occur in equilibrium. This result holds whether agents are computing to improve or refine their values.*

**Proof:** From Theorem 2 it is known that in a single item Vickrey auction with costly computing strong strategic computing can occur in equilibrium. Since the single item Vickrey auction is a special case of the generalized Vickrey auction, it can be concluded that strong strategic computing can also occur in the generalized Vickrey auction.  $\square$

There are additional problems faced by agents participating in the GVA. Agents have multiple valuation problems of their own. If there are  $M$  items then there are  $2^M$  bundles of items in which the agents may place bids. The agents may not have enough computing resources to be able to compute values for each bundle. Instead they must decide on which bundles to *focus their attention*. However, this can not be done in a greedy fashion. It does not work for each agent to simply compute the valuations for bundles that will have the highest utility to the agent in isolation. The agents must take into consideration what bundles other agents will express interest in. Even if agents are not in direct competition for the same bundles of items, they may be interested in overlapping bundles, that is bundles that share some of items. By deliberating on other agents' problems, an agent may be able to determine whether it is likely to win a bundle of items or not. It can then focus its attention on deliberating on the valuation problems for bundles in which it is more likely to be able to win.

In the single item Vickrey auction, if agents had free but limited computing resources then there was no incentive for them to compute on any other problem other than on their own value problem (Theorem 1). However, due to the relations between bundles of goods, strong strategic computing may occur as agents partially evaluate competitors bundles in order to determine their chances of being allocated certain bundles

**Theorem 6** *If agents have anytime algorithms, free but limited computing and their performance profiles are not deterministic then in a generalized Vickrey auction strong strategic computing may occur in equilibrium.*

**Proof:** By example. We will use the performance profile tree representation in this proof for illustrative purposes. Other performance profile representations have the same property. Let there be two agents,  $\alpha$  and  $\beta$ , and three items,  $g_1$ ,  $g_2$ , and  $g_3$ . The performance profiles for the agents' valuation problems are in Figure 8. Valuation problems that remain zero no matter how much computing is allocated to them are not shown.

Assume that both agents' deadlines,  $d_\alpha$  and  $d_\beta$ , occur at  $t = 3$ . Assume, also, that the auction closes at  $T = 3$ . Agent  $\beta$  has a dominant strategy. In the first time step it computes on the valuation for  $\{g_3\}$ . If  $v_\beta^{g_3}(1) = 1.0$  then it computes two more time step on  $\{g_3\}$  to obtain a valuation of 22.0. At time  $T$  it would

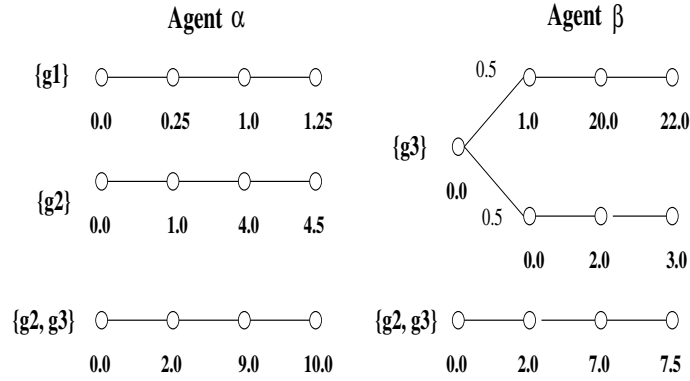


Figure 8: *Performance profiles for agents  $\alpha$  and  $\beta$ . There is uncertainty in agent  $\beta$ 's valuation for bundle  $\{g3\}$ .*

bid its true valuation for all bundles. If  $v_{\beta}^{g3}(1) = 0.0$  then it performs two computing steps on  $\{g2, g3\}$  and obtains a valuation  $v_{\beta}^{\{g2, g3\}}(2) = 7.0$ . At time  $T$  it would bid its true valuation.

Agent  $\alpha$ 's best response is to compute the first time step on agent  $\beta$ 's valuation problem for  $\{g3\}$  (i.e. perform strong strategic deliberation). If after one time step, agent  $\alpha$  determines that  $v_{\beta}^{\{g3\}}(1) = 1.0$  then agent  $\alpha$  computes two time steps on its own valuation problem for  $\{g1\}$ . Otherwise it computes two steps on its own valuation problem for  $\{g2, g3\}$ .

Agent  $\alpha$  realizes that if after one computing step,  $v_{\beta}^{\{g3\}}(1) = 1.0$  then agent  $\beta$  will continue computing on the problem, obtain a valuation of 22.0, and include it in its bid. Since in any optimal allocation, the item  $g3$  will be awarded to agent  $\beta$ , agent  $\alpha$  could never be awarded any bundle that contains  $g3$ . Therefore it is better off computing on the valuation problem for  $\{g1\}$  and bidding its true valuation. However, if  $v_{\beta}^{\{g3\}}(1) = 0.0$  then agent  $\alpha$  knows that it can win the bundle  $\{g2, g3\}$  and so computes two steps on the valuation problem.

The expected utility for each agent is

$$\begin{aligned}
 E[u_{\alpha}] &= \frac{1}{2}(1.0 - 0.0) + \frac{1}{2}(9.0 - 7.0) = 1.5 \\
 E[u_{\beta}] &= \frac{1}{2}(22.0 - 0.0) + \frac{1}{2}(0) = 11.0
 \end{aligned}$$

□

In the single item Vickrey auction, if the performance profiles were deterministic and computing was free but limited, then agents would not even partake of any weak strategic computing. However, in the GVA, this property no longer holds. Agents may at times have incentive to base their computing choices on the deterministic performance profiles of their competitors.

**Theorem 7** *If agents have anytime algorithms, free but limited computing and their performance profiles are all deterministic, then in a generalized Vickrey auction, weak strategic computing may occur in equilibrium.*

**Proof:** By example. Let there be two agents,  $\alpha$  and  $\beta$  and 3 goods,  $g_1, g_2, g_3$ . Assume that the performance profiles of the agents are defined as follows.

$$f_b^\alpha(t) = \begin{cases} 0 & \text{if } t = 0 \\ 4 & \text{if } t > 0 \text{ and } b = \{g_1\} \\ 1 & \text{if } t > 0 \text{ and } b = \{g_2\} \\ 0 & \text{if } t > 0 \text{ and } b = \{g_3\} \\ 0 & \text{if } t > 0 \text{ and } b = \{g_1, g_2\} \\ 6 & \text{if } t > 0 \text{ and } b = \{g_1, g_3\} \\ 0.5 & \text{if } t > 0 \text{ and } b = \{g_2, g_3\} \\ 2 & \text{if } t > 0 \text{ and } b = \{g_1, g_2, g_3\} \end{cases} \quad f_b^\beta(t) = \begin{cases} 0 & \text{if } t = 0 \\ 0 & \text{if } t > 0 \text{ and } b = \{g_1\} \\ 0 & \text{if } t > 0 \text{ and } b = \{g_2\} \\ 0 & \text{if } t > 0 \text{ and } b = \{g_3\} \\ 0 & \text{if } t > 0 \text{ and } b = \{g_1, g_2\} \\ 0 & \text{if } t > 0 \text{ and } b = \{g_1, g_3\} \\ 12 & \text{if } t > 0 \text{ and } b = \{g_2, g_3\} \\ 0 & \text{if } t > 0 \text{ and } b = \{g_1, g_2, g_3\} \end{cases}$$

Assume that the deadline of both agents is at time  $t = 1$ . The agents must decide how to use their single computing step. Agent  $\beta$  has a dominant strategy which is to compute one step on bundle  $\{g_2, g_3\}$  and submit a bid equal to the computed value. If agent  $\alpha$  ignored agent  $\beta$  then it would want to compute on bundle  $\{g_1, g_3\}$  and submit a bid equal to 6. However, agent  $\alpha$  would not be awarded this bundle since it shares an item ( $g_3$ ) with agent  $\beta$ 's desired bundle which will be allocated to agent  $\beta$ . Therefore, agent  $\alpha$  is best off using information from agent  $\beta$ 's performance profile and computing one step on the value problem for bundle  $\{g_1\}$  and then bidding its computed value.  $\square$

## 6. Algorithm Choice

So far in this paper, a basic assumption has been that each agent only has one algorithm for each problem in which it is interested. However, one can imagine situations where agents have access to several different algorithms. In such settings, agents are faced not only with the decision of how to allocate their computing resources across problems, but must also decide how to allocate their computing resources between algorithms for the same problem. It is possible that the choice of algorithm for a problem may depend on what algorithms other agents have chosen and what values that the agents have obtained.

Clearly, in situations where agents incur a cost for computing, strong strategic computing can occur when agents have multiple algorithms for the same problem. However, in settings where agents have limited computing resources, if agents have multiple algorithms, then it is possible that strong strategic computing can occur.

Consider the following example. Assume there are two agents,  $\alpha$  and  $\beta$ , participating in a Vickrey auction. Assume that they both have limited computing resources with deadlines  $D_\alpha$  and  $D_\beta$  and are computing to improve their values. Agent  $\alpha$  has two algorithms for its valuation problem. Algorithm 1 performs well on some types of problems, while Algorithm 2 performs well on other types of problems. However, for the first  $k$  steps, both algorithms perform identically (returning a value of 0) so it is impossible for the agent to determine which is the best algorithm to use until after computing  $k$  steps. Agent  $\beta$  has only one algorithm, but it has the property that it always performs well on problems that Algorithm 1 performs well on, and performs poorly on problems that Algorithm 2 performs well on. An agent using this algorithm of agent  $\beta$  learns after only  $l < k$  steps whether the algorithm is performing well or not. If agent  $\alpha$  only computes using its own algorithms then, in the worst case, it will have to compute  $2k + 1$  steps before it computes a value which is greater than 0. However, by first computing  $l$  steps using agent  $\beta$ 's algorithm, agent  $\alpha$  can determine the type of problem it is facing. It can, therefore, choose the appropriate algorithm for the problem, resulting in a higher expected utility. Using a similar example, it is possible to show that strong strategic computing can also occur if agents computed to refine their values. We obtain the following theorem.

**Theorem 8** *Assume agents have limited computing resources and compute to improve or refine their values. If agents have a choice between multiple algorithms for solving their valuation problems, then strong strategic computing may occur in equilibrium.*

## 7. Related Research

Recently there has been interest in the computer science community on studying computational aspects of mechanism design. 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 computing, the dominant strategy property of Vickrey auctions fails to hold [40]. 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 [30]. 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 compute for the "right" amount of time. A costly computing model, with simple computing 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 [28, 14, 23]. These methods still require that the bidding agents compute and submit their valuations.

In the economics literature there has been some recent work on information acquisition and mechanism design. Perisco [31] studies the incentives to acquire information in first-price and second-price sealed bid auctions. The basic assumptions are that agents have affiliated valuations and that their utility functions satisfy the single-crossing principle. Agents do not know their valuation but can pay some amount to receive a signal which provides information about the valuation of the good. The more they are willing to pay, the more informative the signal is. The game proceeds as follows; Each agent chooses the level of information independently and simultaneously with, the opponent. Then the auction is executed. It is shown that agents choose to acquire more information before a first price action than a second price auction. This is because the value of the item is informative about the opponent's bid since the opponent's signal is positively correlated with the value. This allows a bidder to bid closer to the other bidders, in this first-price auction.

Compte and Jehiel [6] study which auction formats generate better incentives for information acquisition. In their model there are  $n$  bidders, with private valuations drawn from some commonly known distribution, and another bidder,  $a$ , who has only probabilistic information about its valuation, but can pay an amount  $c$  to learn its true valuation.<sup>7</sup> They compare a second-price auction and an ascending auction where the bidder can decide to pay to learn its true valuation at any time. It is shown that there exist situations where the agent will acquire information in the ascending auction but not in the second-price auction.

Rezende [33] studies a dynamic auction where agents are allowed to acquire information about their valuations at any point during the auction. More specifically, the paper studies a Japanese auction with  $n$  bidders who only have probabilistic information about their private valuations, but who can invest some amount  $c_i$  to learn their true valuation. The investment is allowed to take place at any point in time before the close of the auction. Under the assumption that during the auction, the bidders *only* know whether the auction has ended or not (that is, they do not know which, if any, of the other bidders have dropped out), it is shown that in any equilibrium a bidder's best response function has a simple characterization, which is independent of other agents' actions.

Bergemann and Välimäki [3] study general mechanisms. They are interested in mechanisms where agents acquire the efficient amount of information. Like other work in this area, they assume that agents have one chance where they can determine what quality of information they would like to buy (an experiment), where the cost is determined by the preciseness of the information. They define an *ex ante* efficient allocation to be a vector of experiments and an *ex post* efficient allocation such that the vector of experiments maximizes the expected social utility derived from the allocation minus the cost of the experiments. They show that in the private value model then VG mechanism is efficient in this sense. However, in a purely common value setting

---

7. They assume that once the bidder has the item then it learns its true valuation at no cost. They claim that the other situation where the bidder must compute even after getting the item, is analogous.

it is not possible to get both *ex ante* and *ex post* efficiency. That is, agents either over-acquire or under-acquire information.

Rasmusen [32] assumes that agents do not know their valuations but must invest to learn them and are also able to invest in competitors value problems. Like us, he shows that in a second price auction, an agent may base its decision on whether to learn its true valuation on another agent's valuation, but his focus is on understanding behavior such as sniping that is commonly seen in different online auctions like eBay.

## 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 computing 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 computing resources in order to determine their valuations and gain information about the valuations of the other agents participating in the auction.

We investigated agents strategic behavior under different computational settings and auction mechanisms. We showed that in most standard auction settings, in equilibrium agents are willing to use their deliberation resources in order to compute on valuation problems of competitors, even if this means that their knowledge about their own true valuation is lessened. In particular, we showed that strong strategic computing does not depend on the type of anytime algorithm, nor in general does it depend on the type of performance profile. The critical property is whether or not there is uncertainty as to what values will result from further computing. In particular we have shown that if all agents' performance profiles are deterministic then

- No strong strategic computing occurs. This is true for all auctions studied (first-price sealed-bid, Vickrey, ascending and GVA), for both types of anytime algorithms (contract and interruptible) and for both costly and limited computing.
- If computing is costly, then weak strategic computing may occur in all auctions.
- If computing is limited, weak strategic computing may occur in the first-price sealed-bid auction and GVA. Weak strategic computing will not occur in the ascending and Vickrey auctions. This is true for both contract and interruptible algorithms.

If agents' performance profiles are not deterministic (there is uncertainty as to what values will result from computing) then the following results were derived. This is true for both contract and interruptible algorithms.

- If computing is limited then in the ascending and Vickrey auction neither strong nor weak computing occurs in equilibrium. This is true for both contract and interruptible algorithms.
- If computing is limited then in the first-price sealed-bid and GVA strong strategic computing may occur in equilibrium. This is true for both contract and interruptible algorithms.
- If computing is costly, strong strategic computing may occur in all auctions studied. This is true for both contract and interruptible algorithms.

These results are summarized in Table 2.

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.

	Auction mechanism	Counterspeculation by rational agents?	Strategic computation?	
			Limited computation	Costly computation
Single item	Vickrey deterministic performance profile	no	no	weak only
	Vickrey nondeterministic performance profiles	no	no	<b>yes</b>
	Ascending deterministic performance profiles	no	no	weak only
	Ascending nondeterministic performance profiles	no	no	<b>yes</b>
	First Price deterministic performance profiles	yes	weak only	weak only
	First Price nondeterministic performance profile	yes	yes	<b>yes</b>
Multiple items	GVA (deterministic performance profiles)	no	weak only	weak only
	GVA (nondeterministic performance profiles)	no	yes	<b>yes</b>

Table 2: *When does strategic computation occur? The answer to this question depends on the model of limited computation being used.*

## Acknowledgments

This work is based on work supported by the National Research Foundation under CAREER Award IRI-9703122, Grant IIS-9800994, and ITR IIS-0081246. We thank Vincent Conitzer and Michael Wellman for pointing out that if agents have several different algorithms to choose from then strong strategic computing may exist.

## References

- [1] Dilip Abreu and Ariel Rubinstein. The structure of Nash equilibrium in repeated games with finite automata. *Econometrica*, 56(6):1259–1281, 1988.
- [2] Eric B Baum and Warren D Smith. A Bayesian approach to relevance in game playing. *Artificial Intelligence*, 97(1–2):195–242, 1997.
- [3] Dirk Bergemann and Juuso Välimäki. Information acquisition and efficient mechanism design. *Econometrica*, 70:1007–1034, 2002.
- [4] Mark Boddy and Thomas Dean. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67:245–285, 1994.
- [5] E H Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.
- [6] Olivier Compte and Philippe Jehiel. Auctions and information acquisition: Sealed-bid or dynamic formats?, 2001. Working Paper.
- [7] Alan Garvey and Victor Lesser. A survey of research in deliberative real-time artificial intelligence. *Real-Time Systems*, 6:317–347, 1994.

- [8] Itzhak Gilboa and Dov Samet. Bounded versus unbounded rationality: The tyranny of the weak. *Games and Economic Behavior*, pages 213–221, 1989.
- [9] Eric Hansen and Shlomo Zilberstein. Monitoring and control of anytime algorithms: A dynamic programming approach. *Artificial Intelligence*, 126:139–157, 2001.
- [10] Eric Horvitz. Principles and applications of continual computation. *Artificial Intelligence*, 126:159–196, 2001.
- [11] 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. American Association for Artificial Intelligence. Also in L. Kanal, T. Levitt, and J. Lemmer, ed., *Uncertainty in Artificial Intelligence 3*, Elsevier, 1989, pps. 301-324.
- [12] Eric J. Horvitz. Reasoning under varying and uncertain resource constraints. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 111–116, St. Paul, MN, August 1988. Morgan Kaufmann, San Mateo, CA.
- [13] Philippe Jehiel. Limited horizon forecast in repeated alternate games. *Journal of Economic Theory*, 67:497–519, 1995.
- [14] 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.
- [15] 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.
- [16] Richard E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, March 1990.
- [17] Vijay Krishna. *Auction Theory*. Academic Press, 2002.
- [18] 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.
- [19] 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.
- [20] Kate Larson and Tuomas Sandholm. An alternating offers bargaining model for computationally limited agents. In *International Conference on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, July 2002.
- [21] Kate Larson and Tuomas Sandholm. Bidders with hard valuation problems. In *International Conference on Autonomous Agents and Multi-Agent Systems*, pages 160–161, Bologna, Italy, July 2002. Early version: Computationally Limited Agents in Auctions. International Conference on Autonomous Agents 2001, Workshop on Agent-based Approaches to B2B, pp. 27–34, Montreal, Canada, May 28th.
- [22] Kate Larson and Tuomas Sandholm. Miscomputing ratio: The social cost of selfish computing. In *International Conference on Autonomous Agents and Multi-Agent Systems*, Melbourne, Australia, 2003. Early version appeared in the AAAI-02 workshop on Game-Theoretic and Decision-Theoretic Agents, Edmonton, Canada.
- [23] 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.

- [24] Andreu Mas-Colell, Michael Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995.
- [25] Paul Milgrom. Auctions and bidding: A primer. *Journal of Economic Perspectives*, 3(3):3–22, 1989.
- [26] 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.
- [27] John Nash. Equilibrium points in n-person games. *Proc. of the National Academy of Sciences*, 36:48–49, 1950.
- [28] 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.
- [29] Christos H. Papadimitriou and Mihalis Yannakakis. On complexity as bounded rationality. In *STOC*, pages 726–733, 1994.
- [30] 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.
- [31] Nicola Perisco. Information acquisition in auctions. *Econometrica*, 68(1):135–148, January 2000.
- [32] Eric Rasmusen. Strategic implications of uncertainty over one’s own private value in auctions. working paper, January 2003.
- [33] Leonardo Rezende. Mid-auction information acquisition. working paper, Stanford University, November 2002.
- [34] Ariel Rubinstein. *Modeling Bounded Rationality*. MIT Press, 1998.
- [35] Stuart Russell. Rationality and intelligence. *Artificial Intelligence*, 94(1):57–77, 1997.
- [36] Stuart Russell and Devika Subramanian. Provably bounded-optimal agents. *Journal of Artificial Intelligence Research*, 1:1–36, 1995.
- [37] Stuart Russell and Eric Wefald. *Do the right thing: Studies in Limited Rationality*. The MIT Press, 1991.
- [38] Tuomas Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 256–262, Washington, D.C., July 1993.
- [39] Tuomas Sandholm. *Negotiation among Self-Interested Computationally Limited Agents*. PhD thesis, University of Massachusetts, Amherst, 1996. Available at <http://www.cs.cmu.edu/~sandholm/dissertation.ps>.
- [40] 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.
- [41] Herbert A Simon. *Models of bounded rationality*, volume 2. MIT Press, 1982.
- [42] W Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.
- [43] Elmar Wolfstetter. Auctions: An introduction. Technical report, Humboldt University of Berlin, 1994.

- [44] Shlomo Zilberstein, François Charpillet, and Philippe Chassaing. Real-time problem solving with contract algorithms. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1008–1013, Stockholm, Sweden, 1999.
- [45] Shlomo Zilberstein and Stuart Russell. Optimal composition of real-time systems. *Artificial Intelligence*, 82(1–2):181–213, 1996.