

An Algorithm for Computing the Outcome of Combinatorial Auctions with Proxy Bidding

Peter R. Wurman Ashish Sureka Gangshu Cai

North Carolina State University
Department of Computer Science
Raleigh, NC 27695-7535 USA
wurman@ncsu.edu {asureka,gcai}@unity.ncsu.edu

January 13, 2003

Abstract

Combinatorial auctions are difficult to analyze in part because of the vast number of potential strategies available to the bidders. Proxy bidding interfaces limit the users' strategic options by accepting a vector of values and then bidding according to a fixed strategy. However, proxy bidding requires the auctioneer essentially run the auction with the myopic bidders to determine the outcome. In addition to being very costly computationally, this process is only as accurate as the bid increment, and decreasing the bid increment greatly increases the running time. In this paper, we present an algorithm that computes the outcome of the proxy auction by examining only the events that cause the proxy bidders to change their behavior. This algorithm is much faster than actually running the auction, and computes exact solutions.

1 Introduction

Progressive combinatorial auctions (PCAs) have features that make them attractive alternatives to sealed-bid combinatorial auctions like the GVA. However, predicting the outcome of PCAs is difficult in part because of the vast number of conditional strategies available to the bidders. For this reason, several researchers have suggested that bidders interface with the auction system through the use of proxy bidders [1, 4, 6]. In the combinatorial auction setting, these proxy bidders accept a vector of values for the bundles, and represent the user by bidding in a straightforward fashion in a PCA [1, 5, 8].

Restricting interactions with the auction system through proxy agents has several advantages. First, bidding through proxy agents restricts strategic behavior to statements about the relative values of the different combinations. Second, in some situations, proxy bidding reduces the need to estimate the valuations of the other agents because the proxy agent will bid just enough to win, and no more. This is particularly easy to see in a single unit scenario. A bidder in the single unit proxy auction can truthfully state her value, and the proxy agent will bid on her behalf until either her value is reached, or everyone else has dropped out and she wins by paying ϵ more than the second highest bidder. In contrast, a bidder in a sealed-bid, single-unit auction must compute the bid that will maximize her expected utility, which involves estimating the value of the second highest bid. Similarly, in combinatorial auctions, proxy bidding makes it easier for the highest valued coalition to bid just enough to beat the coalition with the second highest value.

Further, proxy bidding represents a way to characterize a spectrum of auction types. At one extreme are sealed-bid auctions in which the bidders submit only one bid. At the other extreme, bidders are allowed to submit a new vector at every iteration, which enables all feasible bidding strategies to be expressed. One of the most desirable features of PCAs is that they enable bidders to avoid determining exact valuations on items that will be irrelevant to the allocation. The sealed bid proxy auction has the undesirable property of requiring the bidder to submit a value for every bundle. However, it seems plausible that proxy auctions with a limited number of proxy messages will restrict the strategic behavior in a useful way and still enable bidders to postpone value determination on some bundles. In this paper, however, we restrict attention to the case where bidders submit a single proxy value. Enhancements to the algorithm to enable multiple proxy statements are briefly discussed at the end of the paper.

To compute the outcome of an auction when proxy bidding is used, the auctioneer essentially runs the auction forward, implementing a well-defined bidding strategy on behalf of each bidder, until bidding stops. This is a very wasteful process and one that can be greatly improved upon. Again, an analogy to a single-unit proxy auction illustrates the point. Consider an auction on eBay in which the reserve price is \$10, and one bidder offered a real bid of \$50. The current price is announced as \$10, and the minimum price increment is \$1. A second bidder now places a bid with a proxy value of \$60. eBay could run an iterative process in which each bidder takes turns out bidding the other until the second bidder bids \$51. Or, eBay could examine the two bids and directly determine that Bidder 2 will beat Bidder 1 and will pay \$1 more than Bidder 1's bid. In this paper we demonstrate that similar computational savings are available in PCAs with proxy bidding. In particular, we present a method to compute the final allocation and prices in a proxy version of the Ascending ($k = 1$)-Bundle Auction (A1BA) [7].

This paper does not address the game theoretic issues that face bidders when selecting a proxy vector to submit to the auction. We concern ourselves solely with the auctioneer's task of computing the outcome given a set of proxy bids.

In the next section we define the proxy version of the A1BA mechanism. Section 3 describes the algorithm that can be used to compute the outcome of the auction.

Section 4 walks through an example of the algorithm. The final two sections contain observations on aspects of the algorithm, a discussion of future work, and the conclusion.

2 Proxy-A1BA

Elsewhere we have introduced A1BA [7] and its proxy-enabled version Proxy-A1BA [8]. We briefly describe them here.

2.1 Model

The auctioneer faces the task of allocating n heterogeneous items to m buyers. Let \mathcal{I} be the set of items, and \mathcal{I} be the set of buyers, and index the sets with j and i , respectively. There are 2^n different combinations of items (including the empty set). Let $\mathbf{b} \in \{0, 1\}^n$ where $b^j = 1$ implies that item j is an element of the bundle \mathbf{b} .

A1BA is a progressive combinatorial auction that produces prices on bundles that are anonymous and support a bundle price equilibrium. Without loss of generality, we assume that buyer i 's bid at time t is expressed as a collection of mutually exclusive offers on bundles of the form $r_i^t(\mathbf{b})$, where $r_i^t(\mathbf{b}) \in \mathcal{R}_+$.¹ In addition, we assume that $r_i^t(\emptyset) = 0$ for all bidders.

A1BA proceeds as follows:

Step 1: Solve the winner determination problem (WDP) expressed in the bids. Let \mathbf{f}^* be the solution to the WDP, and \mathbf{f}_i^* be i 's allocation in \mathbf{f}^* .

Step 2: Construct an assignment (sub)problem that maps the bidders to the set of allocated bundles (plus some dummy bundles for bidders who are winning nothing). Use a pseudo-dual program to find the maximal prices on the assigned bundles that supports the (sub)problem. The pseudo-dual program also computes the surplus, s_i , achieved by each bidder.

Step 3: Set prices on the bundles that were not part of the allocation such that no bidder would achieve greater surplus from an unallocated bundle.

More details on the algorithms are provided elsewhere [7, 8]. Of importance here is the fact that, after each bid, the auction produces a price on each bundle, $\pi_{\mathbf{b}}$, such that

$$\pi_{\mathbf{b}} = \max_i [v_i(\mathbf{b}) - s_i].$$

Bidders interact with the auction system by submitting a bid, v_i that specifies a value for every bundle, \mathbf{b} , denoted $v_i(\mathbf{b})$, to their proxy agent. The proxy agent will then interact in an A1BA mechanism conducted by the auctioneer.

Each proxy agent implements *straightforward bidding*. If it is told by the auction that it is winning c and the prices are π , the agent bids on bundle, \mathbf{b}' , that maximizes

¹This standard XOR format is fully expressive, although not the most concise method of expressing bids [2].

its real surplus at the given prices,

$$\mathbf{b}' = \arg \max_{\mathbf{b}} \begin{cases} v_i(\mathbf{b}) - \pi_{\mathbf{b}} & \text{if } \mathbf{b} = \mathbf{c}, \\ v_i(\mathbf{b}) - (\pi_{\mathbf{b}} + \delta) & \text{otherwise.} \end{cases} \quad (1)$$

where δ is the minimum bid increment. Then, if the solution to (1) provides strictly more surplus than the agent’s tentative allocation, the agent will increase its offer on \mathbf{b}' to $\pi_{\mathbf{b}'} + \delta$.

2.2 Example Problem

One of the advantages of proxy bidding in PCAs not mentioned in the introduction was the ability to run the auction faster. In particular, the proxy agents respond very quickly to price changes and can “immediately” make their next offer. In contrast, bidders with free access to the system will require more time to analyze the information and place a new bid.

Given a set of bids, the auctioneer must now solve the *Proxy Auction Problem* (PAP), that is, it must compute the prices and allocation that results when all of the agents follow the myopic strategy. An algorithm that solves the PAP by iteratively computing incremental bids is *computing by simulation*. However, the simulation approach has several undesirable properties. First, the outcome is dependent upon implementation details such as the tie breaking rule, the order of bidding, and the bid increment. Second, the accuracy is a function of the bid increment. We can improve the accuracy by decreasing the bid increment, but doing so will greatly increase the number of iterations and the amount of time the process takes. This is particularly undesirable because each iteration requires the auction solve an NP-complete WDP and compute new prices.

Fortunately, as in the case with proxy bidding in eBay, it is not necessary to compute the vast majority of bidding interactions in Proxy-AIBA. Moreover, we can compute an exact outcome, one that is not dependent on a bidding increment or tie-breaking rules.

Intuition for the approach can be gained by looking at plots of prices through the course of the proxy auction. Consider the example in Table 1 in which three agents submit bids to the proxy agent for two objects. A graph of the prices computed through simulation² is shown in Figure 1 when the bid increment is set to 0.05. The horizontal axis in the graph represents time and is scaled such that 1 unit = the number of agents divided by the bid increment. With three agents and a bid increment of 0.05, a horizontal unit corresponds to sixty bidding opportunities, and the entire process took 710 iterations.

It is obvious from the figure that, for large periods of time, the prices progress in a steady fashion, and that these periods are punctuated by events that change the rates at which the prices are proceeding. The auction starts off with all three agents bidding on the bundle AB. When $\pi_{AB} = 1$ (at $t = 1/3$), Buyer 1 becomes indifferent between

²The simulation proceeded in a round-robin fashion, with one agent given an opportunity to bid, new prices computed, then the next agent given an opportunity to bid.

	A	B	AB
Buyer 1	8	7	9
Buyer 2	1	3	9
Buyer 3	2	1	10

Table 1: An example with three buyers bidding on two objects.

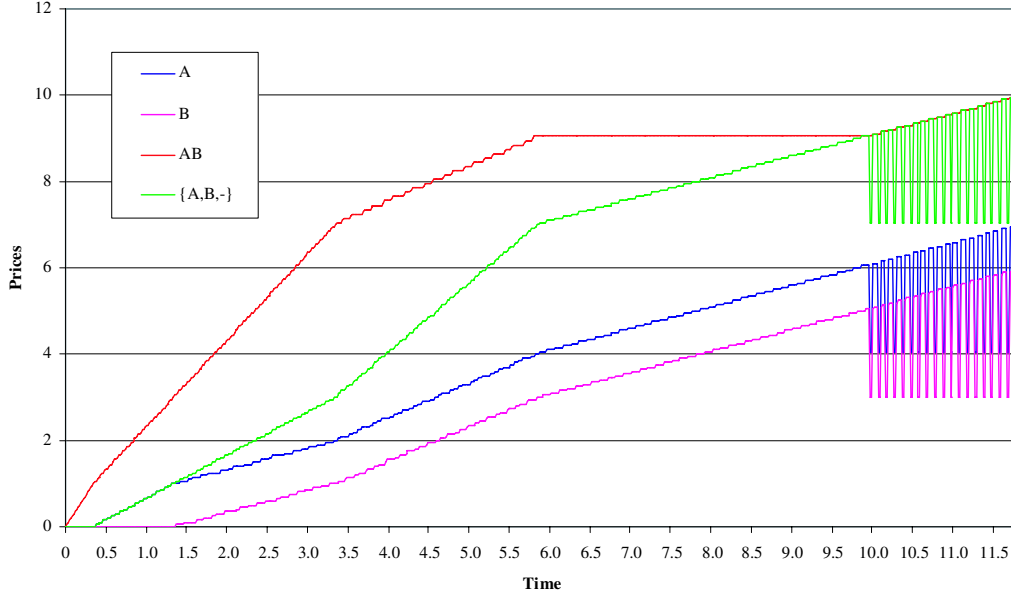


Figure 1: The prices over time when proxy agents bid on the example in Table 1 with bid increment 0.05. The prices of A, B, and AB, and the value of the allocation in which Buyer 1 gets A and Buyer 2 gets B are plotted.

A and AB, and in fact begins to bid on A while Buyers 1 and 2 continue bidding on only AB. When the price of $\pi_{AB} = 2$, Buyer 1 becomes willing to bid on B. However, Buyer 1 does not abandon A. Instead, she alternates between bidding on A and bidding on B, and both prices proceed in parallel. A few turns later, Buyer 2 begins to bid on B when the difference in price between AB and B reaches 6.0. At that point, Buyer 2 splits time in some fashion between the two bundles. At the same time, Buyer 3 competes with Buyer 2 by bidding on AB when necessary, and passing whenever he is already winning. These decisions affect the slope of the price of AB, and similar explanations exist for all of the inflection points evident in the graph, including the extended period during which the price of AB does not change. The oscillations at the end occur when two allocations are competing, one in which Buyer 3 gets AB, and one in which Buyer 1 gets A and Buyer 2 gets B. The prices oscillate because when the second allocation is winning, the prices of A and B must be set such that Buyer 2 wants B, and Buyer 1 wants A.

In the next section, we show that it is possible to compute the final allocation and prices by analyzing only the inflection points. We will refer to the above example

throughout the discussion of the algorithm.

3 An Algorithm for PAP

The algorithm for predicting prices is loosely based upon a metaphor of particle trajectories. Rather than model bidding behavior explicitly, we construct a model of the price trajectories and keep track of each agent’s contribution toward the trajectory. Changes in trajectory roughly correspond to points in time at which particle collisions occur. Due to the limitations of space, we sketch the algorithm but do not go into full detail. In particular, we will simplify the scenario by assuming than collisions occur one at a time. The implications of relaxing this assumption is addressed in Section 5.

3.1 Behavioral Implications of Proxy Bidding

The algorithm is based on the following three observations that are directly implied by the myopic bidding strategy employed by the proxy agent.

1. A proxy agent will pass if it is currently winning or if the prices are all greater than or equal to its valuations.
2. A proxy agent will begin bidding on an item that was not attractive when the price difference between that item and those to which it is attracted becomes the same as the difference in its valuation for those items.
3. A proxy agent will stop bidding on an item if the item’s price is increasing faster than other items that the agent is attracted to.

3.2 Bundle-Price Trajectory Algorithm

Before describing the algorithm, we need to define several concepts. At a given time, each agent has a *demand set*, D_i^t , that is the set of bundles the agent would be willing to purchase at the prices announced at time t . The algorithm partitions time into *intervals* during which the demand sets for all agents remain constant. Intervals have duration, Δt . We denoted intervals $\tau_{t,\Delta t} \equiv [t, t + \Delta t)$. When it is not confusing, we identify the interval simply by its starting time t .

3.2.1 Agent Attention

Each agent has a certain amount of *attention* that it can allocate between the bundles. Attention can be thought of as the proportion of the agent’s time that will be spent bidding on a particular bundle. Let $\theta_{i,b}^t$ be the amount of attention per unit time that i is giving to b over the interval beginning at t , and let $\theta_{i,\text{pass}}^t$ be the proportion of time the agent will not bid because it is already winning. Each agent has a total of one unit of attention that can be allocated between passing and bidding on bundles in its

demand set. That is, at time t ,

$$\theta_{i,\text{pass}}^t + \sum_{\mathbf{b} \in D_i^t} \theta_{i,\mathbf{b}}^t = 1.$$

The allocation of attention between bundles is constant through the interval.

The *trajectory* of the price of \mathbf{b} in the interval is the sum of the attention being paid to \mathbf{b} . Let $\theta_{\mathbf{b}}^t$, the trajectory of bundle \mathbf{b} in the interval beginning at t , be computed as

$$\theta_{\mathbf{b}}^t = \sum_{\{i | \mathbf{b} \in D_i^t\}} \theta_{i,\mathbf{b}}^t.$$

With the current price and trajectory of each bundle, it is possible to directly compute the next point in time at which an inflection in the bundle's trajectory will occur due to the relationship between prices and agent's valuations. Consider an agent i , and two bundles, \mathbf{b} and \mathbf{c} , where \mathbf{b} is in D_i^t and \mathbf{c} is not. The fact that \mathbf{c} is not in D_i^t implies that $v_i(\mathbf{b}) - \pi_{\mathbf{b}}^t > v_i(\mathbf{c}) - \pi_{\mathbf{c}}^t$. At what point, if any, will i become interested in \mathbf{c} ? The agent will become interested if and when the inequality becomes an equality. We call the point at which that happens a *collision*.³ The collision point can be computed from the true bid, the prices, and the trajectories on the prices. Let $\Delta t_i^{\mathbf{b},\mathbf{c}}$ be the time increment from the present at which the collision between \mathbf{b} and \mathbf{c} occurs for agent i . If \mathbf{b} and \mathbf{c} collide, then at time $\Delta t_i^{\mathbf{b},\mathbf{c}}$ into the future,

$$v_i(\mathbf{b}) - (\pi_{\mathbf{b}}^t + \theta_{\mathbf{b}}^t \Delta t_i^{\mathbf{b},\mathbf{c}}) = v_i(\mathbf{c}) - (\pi_{\mathbf{c}}^t + \theta_{\mathbf{c}}^t \Delta t_i^{\mathbf{b},\mathbf{c}}).$$

Solving for $\Delta t_i^{\mathbf{b},\mathbf{c}}$ gives

$$\Delta t_i^{\mathbf{b},\mathbf{c}} = \frac{v_i(\mathbf{b}) - \pi_{\mathbf{b}}^t - v_i(\mathbf{c}) + \pi_{\mathbf{c}}^t}{\theta_{\mathbf{b}}^t - \theta_{\mathbf{c}}^t}. \quad (2)$$

Equation (2) establishes the time increment over which an agent would not change its bid on a particular bundle, \mathbf{b} , in order to pursue another bundle, \mathbf{c} . Note that the denominator of 2 could be zero, or $\Delta t_i^{\mathbf{b},\mathbf{c}}$ could be negative, both of which indicate that the two bundles do *not* collide in the future, and, *ceteris paribus*, i will not be attracted to \mathbf{c} . Note also that the empty bundle is always evaluated as a potential \mathbf{c} . Collisions with the empty bundle are special because, when the agent becomes attracted to \emptyset , doing nothing now has the same value to her as bidding. From that point on, the agent becomes *inactive* and will not bid again. Let C_i^t be the set of bundles not in D_i^t for which $\Delta t_i^{\mathbf{b},\mathbf{c}} > 0$.

To establish the duration of the interval starting at time t , we need the first collision point among all agents; the trajectories will change at the first collision, rendering the other computed collisions obsolete. Thus, the duration of time until the next bundle collision is

$$\Delta t^{\text{BP}} = \min_i \left\{ \min_{\mathbf{b} \in D_i^t, \mathbf{c} \in C_i^t} \Delta t_i^{\mathbf{b},\mathbf{c}} \right\} \quad (3)$$

³In terms of the particle metaphor, it is collision between the particles that represent the surpluses offered by the two bundles.

Let $\hat{\mathbf{b}}$ and \hat{i} be the bundle and agent, respectively, that force (3) to be satisfied. Here we make the simplifying assumption that $\hat{\mathbf{b}}$ and \hat{i} uniquely satisfy (3).⁴ The interpretation of this information is that Δt represents the amount of time before \hat{i} would add $\hat{\mathbf{b}}$ to its demand set. That is $\hat{\mathbf{b}} \in D_{\hat{i}}^{t+\Delta t}$.

3.2.2 Competitive Allocations

Bidding behavior is also influenced by whether or not an agent is part of a winning coalition. If the agent is not winning, and has any surplus left to commit, it will bid again. While the agent is winning, it will not bid because it is already in a state in which the object it is winning (myopically) presents the greatest surplus at the given prices (i.e., the first condition of 1 is satisfied).

To capture this dynamic, we introduce the concept of a *competitive allocation*. Recall that \mathbf{f} is a solution to the allocation problem, that is, $\mathbf{f} : \mathcal{J} \rightarrow \mathcal{I}$. Let F be the set of all feasible allocations, and $\mathbf{f} \in F$. It is convenient to abbreviate an allocation using an ordered set where the position corresponds to the agent and the value corresponds to the bundle. For example, $\{A, B, -\}$ indicates that agent 1 receives A, agent 2 receives B, and agent 3 receives nothing.

In addition to the trajectories of the bundle prices, the algorithm maintains information about the trajectories and values of the feasible allocations.⁵ This information allows the algorithm to determine when the set of competitive allocations will change, causing a change in agent behavior.

A feasible allocation specifies a set of agents, \mathcal{I}_+ , a set of assigned bundles, \mathcal{B}_+ , and a mapping between the two. Let \mathbf{f}_i be the assignment to i in solution \mathbf{f} . To compute the value of a solution at a given point in time, we need to distinguish between those agents actively bidding on \mathbf{f} , and those that are not. Let $W_{\mathbf{f}}^t$ be the set of agents for which $\mathbf{f}_i \in D_i^t$, and $Z_{\mathbf{f}}^t = \mathcal{I}_+ \setminus W_{\mathbf{f}}^t$.

For those agents not actively bidding on the allocation at time t , we need to know their contribution to the value of the solution. Let

$$q(i, \mathbf{f}_i) \equiv \begin{cases} 0 & \text{if } i \text{ has not bid on } \mathbf{f}_i, \\ \pi_{\mathbf{f}_i}^t & \text{where } t \text{ is the most recent time} \\ & \text{at which } \mathbf{f}_i \text{ was removed from } D_i^t. \end{cases}$$

We can now specify the value of an allocation at time t :

$$V^t(\mathbf{f}) = \sum_{i \in W_{\mathbf{f}}^t} \pi_{\mathbf{f}_i}^t + \sum_{i \in Z_{\mathbf{f}}^t} q(i, \mathbf{f}_i).$$

⁴The assumption doesn't restrict the types of problems that can be solved, but it does simplify management of the demand sets by avoiding situations that require more complex and general procedures engendered by putting multiple items into play at the same time.

⁵As a practical matter, the algorithm does not need to track all n^m possible feasible allocations. Instead, it can track only those that have positive value. Moreover, we hypothesize that in the worst case, the algorithm need track only a single best allocation for any given coalition of agents.

f is a competitive allocation at time t if

$$V^t(f) = \max_{\hat{f} \in F} V^t(\hat{f}) \quad (4)$$

That is, f is competitive if it is among those that generate the maximal value. If (4) does not hold, then f is not competitive. Let Φ be the set of competitive allocations.

Agents that are involved in bidding for competitive allocations will increase their bid only when their allocation is beaten, and pass otherwise. Further, agents that are not part of any feasible allocation will give their full attention to the bundles in their demand set and will not pass at all. We will see that these behaviors contribute to the computation of new trajectories in Section 3.3. However, before addressing the algorithm, we need to consider the effect that competitive allocations have on computing the intervals.

In Figure 1, we see that at time $t = 5\frac{5}{6}$, π_{AB} remains flat for six time intervals. Then, at $t' = 11\frac{2}{3}$, it begins increasing. This behavior occurs because at π_{AB}^t , Buyer 3 became the only bidder on AB, and the solution $\{-, -, AB\}$ became the only competitive allocation. Thus, Buyer 3 does not need to increase his bid because he has no competition. This remains true until t' , at which time the line corresponding to the allocation $\{A, B, -\}$ becomes competitive, forcing Buyer 3 to resume bidding. Thus, we need a method to compute the collisions of currently non-competitive allocations with competitive ones.

First, recognize that the trajectory of an allocation is the sum of the trajectories of the components that are actively being bid upon. Let γ_f^t be the trajectory of f , computed as

$$\gamma_f^t = \sum_{i \in W_f^t} \theta_{f_i}^t.$$

Competitive allocation, f , will collide with currently non-competitive allocation, \hat{f} , when the values of the two become equal. If the current time is t , the collision will occur in $\Delta t^{f, \hat{f}}$ time increments, where

$$\Delta t^{f, \hat{f}} = \frac{V^t(\hat{f}) - V^t(f)}{\gamma_f^t - \gamma_{\hat{f}}^t}. \quad (5)$$

As in the computation of collisions in Section 3.2.1, a non-positive value for $\Delta t^{f, \hat{f}}$ means that, with the current allocation of attention, \hat{f} will not become competitive with f . Finally,

$$\Delta t^{\text{CA}} = \min_{f \in \Phi, \hat{f} \notin \Phi} \Delta t^{f, \hat{f}}. \quad (6)$$

3.3 Bundle Price Trajectory Algorithm

We now have the foundation to describe the algorithm. At a high level, the algorithm is quite simple. We initialize the algorithm by setting all prices to zero, then enter the following loop:

1. Compute the demand set for each agent.
2. Compute the allocation of attention by each agent.
3. Compute the competitive allocations.
4. Compute the duration of the interval, or terminate.
5. Compute the prices at the end of the interval.

We elaborate on these steps below. However, for expository reasons we explain step 2 before step 1.

3.3.1 Computing the Allocation of Attention

The allocation of attention between the bundles in the demand set at time t can be computed using the following linear program.

$$\begin{aligned}
\min \quad & \sum_i \theta_{i,\text{pass}}^t & (7) \\
\text{s.t.} \quad & \sum_i \theta_{i,\mathbf{b}}^t = \sum_i \theta_{i,\mathbf{c}}^t, \quad \forall i, \{\mathbf{b}, \mathbf{c}\} \in D_i^t, \\
& \sum_{i \in \mathbf{f}} \theta_{i,\mathbf{f}_i}^t = \sum_{i \in \hat{\mathbf{f}}} \theta_{i,\hat{\mathbf{f}}_i}^t, \quad \forall \{\mathbf{f}, \hat{\mathbf{f}}\} \in \Phi, \\
& \theta_{i,\text{pass}}^t + \sum_{\mathbf{b}} \theta_{i,\mathbf{b}}^t = 1, \quad \forall i, \\
& \theta_{i,\mathbf{b}}^t, \theta_{i,\text{pass}}^t \geq 0, \quad \forall i.
\end{aligned}$$

The objective function is to minimize the amount of time agents pass. The first constraint is imposed by an agent's desire to equilibrate two bundles that are producing the same surplus, and ensures that, if \mathbf{b} and \mathbf{c} are part of the demand set, they will have the same trajectory. The second constraint is imposed by the need for competitive allocations to compete, and requires that, if \mathbf{f} and $\hat{\mathbf{f}}$ are feasible, competitive allocations, the trajectories of the allocations will be identical. The final constraint simply ensures that each agent allocate one unit of attention.

There are two special cases. First, if there is only one competitive allocation, say \mathbf{f} , then the second constraint will be written as

$$\sum_{i \in \mathbf{f}} \theta_{i,\mathbf{f}_i}^t = 0.$$

This ensures that the agents in the single competitive allocation do not bid (until another allocation becomes competitive).

Second, if the bundle being added to the demand set is the empty bundle, then the agent stops bidding. Deactivated agents are accounted for by setting $\theta_{i,\text{pass}}^t = 1$.

3.3.2 Computing the Demand Set

Initially, the demand set of each agent is the set of bundles that maximizes her surplus when the prices are zero. Thereafter, the demand set is updated at each intersection. There are two cases to consider: the triggering event was caused by a bundle collision or by a non-competitive allocation becoming competitive.

Recall that we have made the simplifying assumption that $\hat{\mathbf{b}}$ and \hat{i} are the bundle and agent that uniquely satisfy (3). Obviously, $\hat{\mathbf{b}}$ will be added to the demand set of agent i . However, under some conditions, we must remove some bundles that were previously in D_i^t . Intuitively, we must remove a bundle, \mathbf{b} , from i 's demand set when \mathbf{b} 's trajectory will be greater than that of $\hat{\mathbf{b}}$; if the price of \mathbf{b} increases faster than the price of $\hat{\mathbf{b}}$, \mathbf{b} will become relatively expensive and i will stop bidding on it.

We can determine whether $\mathbf{b} \in D_i^{t-1}$ should remain in D_i^t by evaluating the solution to the problem where agent i excludes \mathbf{b} from consideration. To do this, we solve (7) without \mathbf{b} in i 's demand set. If the solution to this altered problem results in a trajectory for \mathbf{b} that is greater than the trajectory for $\hat{\mathbf{b}}$, then \mathbf{b} should be removed from i 's demand set.⁶

3.3.3 Computing the Competitive Allocations

The set of competitive allocations can be computed directly using the procedure outlined in Section 3.2.2.

3.3.4 Compute the Interval, or Terminate

Having computed the competitive allocations, we now check whether the algorithm should terminate. The algorithm terminates when there is only one competitive allocation and the trajectories of all other allocations are zero (because their agents are inactive).

If the termination conditions are not met, we compute the interval over which the current allocation of attention applies. This can be done by taking the smaller of Δt^{BP} and Δt^{CA} .

3.3.5 Compute the New Prices

Finally, we update the prices. The price of \mathbf{b} at time $t + \Delta t$ is simply

$$\pi_{\mathbf{b}}^{t+\Delta t} = \pi_{\mathbf{b}}^t + \Delta t \theta_{\mathbf{b}}^t. \quad (8)$$

However, Figure 1 suggests that the picture might be more complicated than (8) suggests. In particular, at the end of the auction, the prices of A and B are oscillating. The oscillation is caused by the alternating competing allocations. When Buyer 3 is winning AB, the prices of A and B are set to Buyer 1's bids. However, when

⁶We have described our current best method for determining whether to remove a bundle from the demand set when collisions are unique. We have made some progress on generalizing the concept to the case where multiple agents are updating their demand set, but do not yet have a solution for all cases.

Buyer 1 increases her bid and the allocation $\{A, B, -\}$ becomes the winner, the A1BA auction sets the prices such that Buyer 1 does not want the bundle allocated to Buyer 2. Because Buyer 2 stopped bidding when $\pi_B = 3$, A1BA will determine that the price of A must be 4 in order for Buyer 1 to not prefer to buy B.

The algorithm need not be changed to accommodate this aspect of the A1BA mechanism. The prices computed according to (8) are consistent with the price that an agent will have to offer when it is *not* winning, which is precisely the time at which she has the impetus to bid. The lower prices represent the case when the agent is winning, in which case she has no reason to bid. In some sense, the upper edge of this envelope corresponds to the *ask price*, and the lower edge to the *bid price*. An agent who is not winning must make an offer above the ask price. However, the auction will clear at the bid price. In the algorithm, we can postpone computing the final bid price until the termination conditions are met.

4 Worked Example

We now apply the algorithm to the example in Table 1. At $t = 0$, the prices are zero and AB is the sole element in each agent’s demand set. There are three competitive allocations corresponding to giving AB to each of the three agents. The solution to the attention allocation problem is $\theta_{i,AB}^0 = 1$.

Table 2 shows the remaining steps of the algorithm. The table shows the demand set and the attention values for each agent in each step. The “set minus” notation is used in the demand set column to indicate a bundle that is identified to be removed. The rightmost two columns show the positively valued allocations and their values (the correspondence between the rows of buyers and the allocations is irrelevant), with competitive allocations marked with an asterisk. In the first two intervals, the allocation $\{A, ?, ?\}$ represents the family of solutions in which B could be assigned to either Buyer 2 or 3.

A careful comparison between the trajectories implied in Table 2 and the actual prices computed by simulation shows a near exact correspondence. In fact, the algorithm computed a more accurate result than the simulation, which took discrete steps of size 0.05 and 710 iterations.

A particularly interesting step occurs at $t = 3\frac{1}{3}$. Notice the unexpected distribution of attention, with Buyer 1 bidding on both A and B, Buyer 2 bidding on B and AB, and Buyer 3 bidding on AB but only with attention 0.4. Each agent has one unit of attention allocated, and all three price trajectories have a slope of 0.8 (notice the correspondence with this region of Figure 1). Further, Buyer 2 and Buyer 3 are putting equal attention towards AB because they represent the two competitive allocations.

5 Discussion and Future Work

Although space precludes much discussion, there several issues worth mentioning briefly. First, we have presented this novel algorithm with intuitive

explanations of why it works drawn from basic properties of the proxy agent's bidding behavior. As part of the future effort, we will attempt to achieve the more satisfactory result of proving that the algorithm computes the outcome of the simulation as the simulation's bid increment goes to zero. This work will also include an analysis of the computational complexity of the algorithm.

In addition, there are a variety of possible methods for improving the performance of the algorithm. In particular, we described the interval computations as a comparison between all combinations of elements in a set with those not in the set. With some clever bookkeeping, a much smaller number of comparisons need to be made. We also plan to generalize the process of managing the demand set so that the single-collision assumption can be lifted.

Third, we expect that the approach has applicability outside of Proxy-A1BA. It can be used with little modification in auctions that allow multiple proxy statements. In addition to Proxy-A1BA, the procedure captures the behavior of a pay-your-bid auction with proxy bidders by simply not computing the equilibrium prices at the end. The approach may also suggest methods for computing the outcome of Ausubel and Milgrom's Ascending Proxy Auction [1], and perhaps, *iBundle* [3].

6 Conclusion

In this paper, we present a novel algorithm for computing the outcome of proxy bidding in the A1BA mechanism. This method is far superior to the straightforward process of simulating the auction both in computational time and solution accuracy. This work opens the door for many interesting avenues of related research, including adapting the approach for other combinatorial auctions with proxy bidding.

Acknowledgments

This research was funded by NSF CAREER award 0092591-0029728000, the Defense Advanced Research Projects Agency under contract F30602-99-1-0540, and the E-Commerce program NC State. The views and conclusions contained herein are those of the authors, as are any errors.

References

- [1] L. M. Ausubel and P. R. Milgrom. Ascending auctions with package bidding. *Frontiers of Theoretical Economics*, 1(1):1–42, 2002.

- [2] N. Nisan. Bidding and allocation in combinatorial auctions. In *Second ACM Conference on Electronic Commerce*, pages 1–12, 2000.
- [3] D. C. Parkes. *iBundle*: An efficient ascending price bundle auction. *First ACM Conference on Electronic Commerce*, pages 148–157, November 1999.
- [4] D. C. Parkes and L. H. Ungar. Preventing strategic manipulation in iterative auctions: Proxy agents and price-adjustment. In *Seventeenth National Conference on Artificial Intelligence*, 2000.
- [5] P. Phadke. An evolutionary approach to finding bidding strategies in a combinatorial auction. Master’s thesis, North Carolina State University, 2002.
- [6] J. Rodríguez, P. Noriega, C. Sierra, and J. Padget. FM96.5 a java-based electronic auction house. In *Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM ’97)*, 1997.
- [7] P. R. Wurman and M. P. Wellman. *AkBA*: A progressive, anonymous-price combinatorial auction. In *Second ACM Conference on Electronic Commerce*, pages 21–29, 2000.
- [8] P. R. Wurman and M. P. Wellman. Nonlinear pricing in combinatorial auctions. Technical report, North Carolina State University, November 2002.

$t = \frac{1}{3}$ $\pi_A = 0$ $\pi_B = 0$ $\pi_{AB} = 1$								
Agent	C_i	D_i	$\theta_{i,A}^t$	$\theta_{i,B}^t$	$\theta_{i,AB}^t$	$\theta_{i,pass}^t$	\mathbf{f}	$V(\mathbf{f})$
Buyer 1	A	\ AB	1			0	{A, ?, ?}	0
Buyer 2		AB			1	0	{-, AB, -}	1*
Buyer 3		AB			1	0	{-, -, AB}	1*
$t = 1\frac{1}{3}$ $\pi_A = 1$ $\pi_B = 0$ $\pi_{AB} = 3$								
Agent	C_i	D_i	$\theta_{i,A}^t$	$\theta_{i,B}^t$	$\theta_{i,AB}^t$	$\theta_{i,pass}^t$	\mathbf{f}	$V(\mathbf{f})$
Buyer 1	B	A	.5	.5		0	{A, ?, ?}	1
Buyer 2		AB			1	0	{-, AB, -}	3*
Buyer 3		AB			1	0	{-, -, AB}	3*
$t = 3\frac{1}{3}$ $\pi_A = 2$ $\pi_B = 1$ $\pi_{AB} = 7$								
Agent	C_i	D_i	$\theta_{i,A}^t$	$\theta_{i,B}^t$	$\theta_{i,AB}^t$	$\theta_{i,pass}^t$	\mathbf{f}	$V(\mathbf{f})$
Buyer 1		A, B	.8	.2		0	{A, B, -}	3
Buyer 2	B	AB		.6	.4	0	{-, AB, -}	7*
Buyer 3		AB			.4	.6	{-, -, AB}	7*
$t = 5\frac{5}{6}$ $\pi_A = 4$ $\pi_B = 3$ $\pi_{AB} = 9$								
Agent	C_i	D_i	$\theta_{i,A}^t$	$\theta_{i,B}^t$	$\theta_{i,AB}^t$	$\theta_{i,pass}^t$	\mathbf{f}	$V(\mathbf{f})$
Buyer 1		A, B	.5	.5		0	{A, B, -}	7
Buyer 2	null	AB \ B				1		
Buyer 3		AB				1	{-, -, AB}	9*
$t = 9\frac{5}{6}$ $\pi_A = 6$ $\pi_B = 6$ $\pi_{AB} = 9$								
Agent	C_i	D_i	$\theta_{i,A}^t$	$\theta_{i,B}^t$	$\theta_{i,AB}^t$	$\theta_{i,pass}^t$	\mathbf{f}	$V(\mathbf{f})$
Buyer 1		A, B	.5	.5		0	{A, B, -}	9*
Buyer 2		null				1		
Buyer 3		AB			.5	.5	{-, -, AB}	9*
$t = 11\frac{5}{6}$ $\pi_A = 7$ $\pi_B = 7$ $\pi_{AB} = 10$								
Agent	C_i	D_i	$\theta_{i,A}^t$	$\theta_{i,B}^t$	$\theta_{i,AB}^t$	$\theta_{i,pass}^t$	\mathbf{f}	$V(\mathbf{f})$
Buyer 1		A, B	.5	.5		0	{A, B, -}	10*
Buyer 2		null				1		
Buyer 3	null	\ AB				1	{-, -, AB}	10

Table 2: The results of the computations when the algorithm is applied to the example in Table 1.