Computing Price Trajectories in Combinatorial Auctions with Proxy Bidding

Jie Zhong
Gangshu Cai
Peter R. Wurman

North Carolina State University

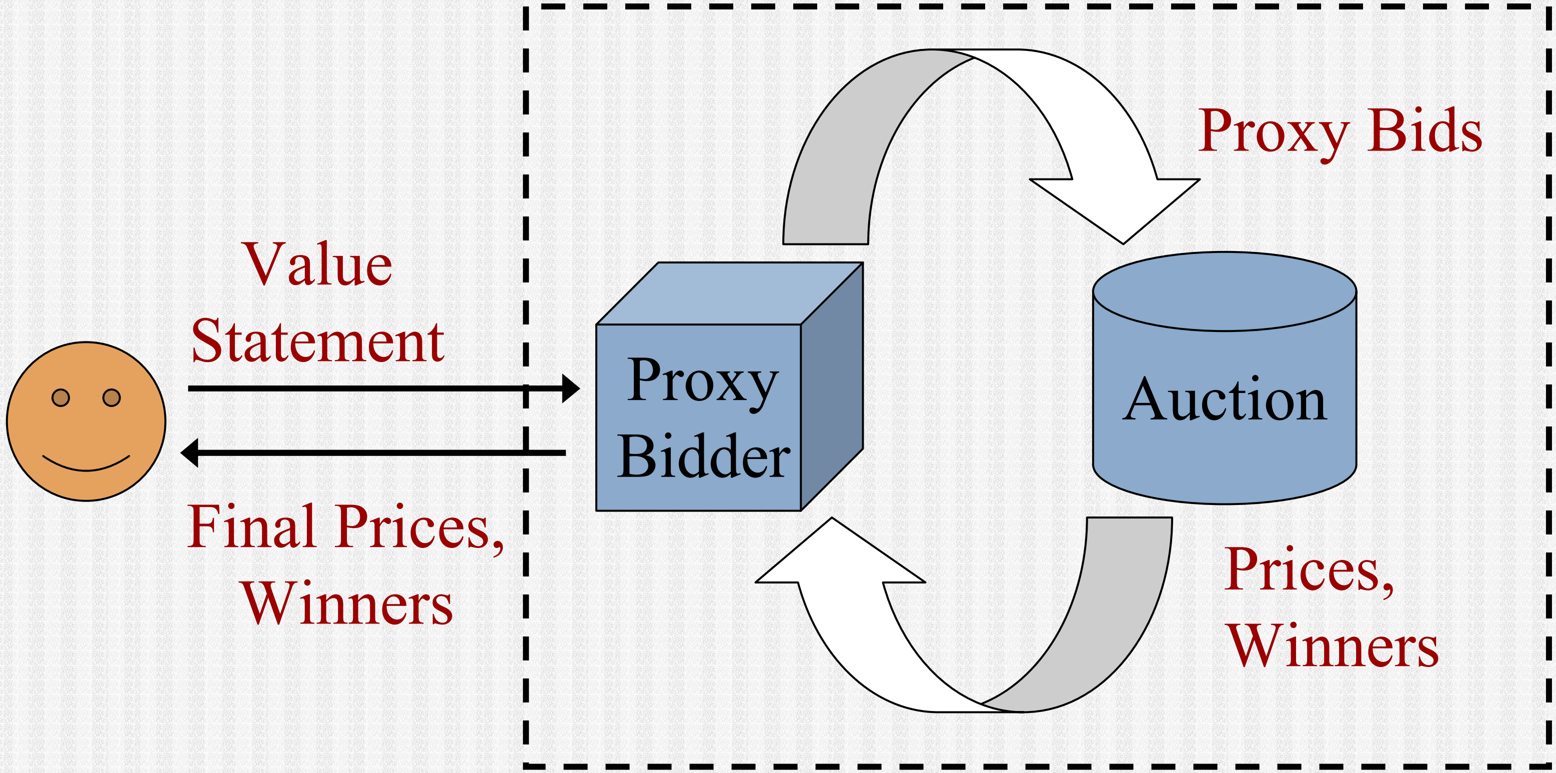
Overview

- Problem Definition
- Intuition
- The Algorithm
- Conclusion

Proxy Bidding in Combinatorial Auctions

- Bidders give a set of values to an agent
- Agents place bids in an internal auction that solves the WDP and announces prices

Proxy Bidding Diagram



Benefits

- Speeds up auction
- Simplifies the strategy space
- Interactions with proxies may have several steps, allowing deferred computation of valuations

A Simple Iterative Combinatorial Auction

- Bidders make offers on bundles of items
- All bids are retained
- Price bundles at highest bid
- Inform current winners
(not necessarily the highest bidders)
- Non-winning bidders must beat price by δ

* this will *not* be a strategic analysis!

Proxy Bidding Rules

- If the agent is not already winning something, it bids on the item that provides the most surplus

$$b^* = \arg\max_b \{v_i(b) - \mathrm{p}_b\}$$

where p_b is the price of bundle *b.*

- Bid $\mathrm{p}_b + \delta$
- If more than one *b* satisfies, then randomly select one.

Example

	A	B	AB	C	AC	BC	ABC
a_1	10	3	18	2	18	10	20
a_2	4	9	15	3	12	18	20
a_3	1	3	11	9	16	17	25
a_4	7	7	16	7	16	16	20

The Proxy Auction Problem

- PAP: Compute the final prices and allocation of a proxy auction given the bids
- By Simulation
  - Agents bid
  - WDP and prices are computed
  - Repeat

Simulation is Undesirable Because…

- Accuracy depends on bid increment
- Slow: Solves multiple WDPs
- Sensitive to magnitude of values
- Sensitive to ordering of agents
- Sensitive to tie-breaking rules

- There is some regularity that we can take advantage of…

Some Observations

- Periods of steady progress
  - Agents maintain a demand set
  - Spread bids among bundles in demand set
- Punctuated by changes in behavior when
  - A new bundle is added to someone's demand set
  - An agent drops out
  - An allocation becomes competitive and its members start passing

The Algorithm: Key Concepts

D_i^t - Demand Set

- The bundles that give an agent the maximal surplus at current prices.

$\theta_{i,b}^t$ - Attention

- The proportion of time an agent spends bidding on a bundle in its demand set.

θ_b^t - Trajectory

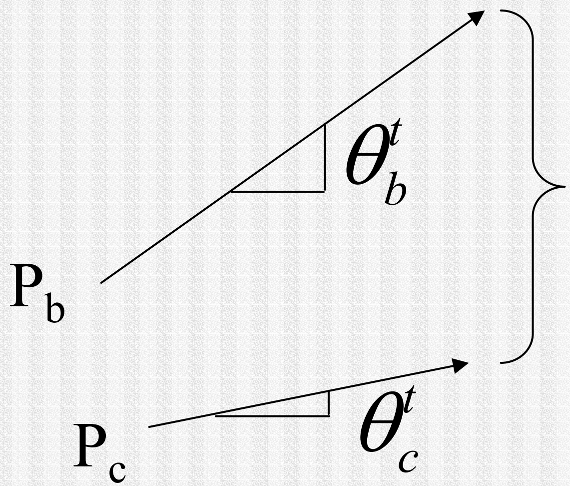
- The slope of the price of b, $\theta_b^t = \sum_i \theta_{i,b}^t$

Competitive Allocations

- The set of *competitive allocations* (CAs) contains the solutions, *f*, with the maximal value, i.e., $V(f) = \max_{\hat{f}} V(\hat{f})$
  - Must account for bidders who are actively bidding and those who have stopped bidding
- CAs have slopes: $\theta_f = \sum_{i \in f} \theta_{i,f_i}$
- CAs are winning with frequency β_f

New Bundle Collisions

- For $b \in D_i^t, c \notin D_i^t$



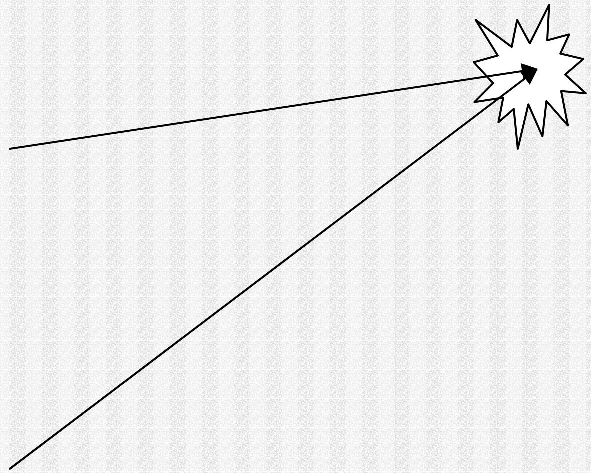
$$v_i(b) - p_b = v_i(c) - p_c$$

- When the surplus that *i* gets from *c* is as good as from *b*, *i* will add *c* to its demand set
- Special case: when the null bundle enters demand set, agent becomes inactive

Competitive Allocation Collisions

- For $f \in \mathrm{CA}, \hat{f} \notin \mathrm{CA}$

$$f = \{-,-,AB\}$$



$$\hat{f} = \{A,B,-\}$$

Computing the Duration of an Interval

- The *interval* is the amount of time until the next collision
  - Compute the earliest surplus collision(s)
  - Compute the earliest CA collision(s)
  - Select the min

At a Collision

- When a collision occurs
  - Some bundles may leave demand sets
  - Some allocations may no longer be competitive
- Thus, we know the *potential* demand sets and *potential* CAs, but not which will remain so in the next interval

Solving the Allocation of Attention, Demand Sets, & CAs

$$\max \sum x_f \qquad \text{s.t.}$$

$$\theta_b - \theta_c + N y_{i,b} + N y_{i,c} \le 2N$$

$$\theta_b - \theta_c + N y_{i,b} - N y_{i,c} \le N$$

$$\sum \beta_f = 1$$

$$\theta_{\hat{f}} + N x_f + N x_{\hat{f}} \le 2N$$

$$\theta_{\hat{f}} - N x_f + N x_{\hat{f}} \le N$$

$$+ \sum \theta_{i,b} = K_i$$

$$_{i,pass} = \sum (1 - G_{f,i}) \beta_f$$

$$\le x_f \le \sum y_{i,f_i}$$

$$\le y_{i,b} \le N\theta_b$$

Integer Variables:

y

When b is in i's demand set

The sum of the frequency with which CAs are selected

When f is competitive, if $\hat{f}$ is also, then their slopes are equal, otherwise the slope of f is greater than $\hat{f}$

Solving the Allocation of Attention, Demand Sets, & CAs

$$\ldots + Ny_{i,b} + Ny_{i,c} \le 2N$$

$$\ldots - Ny_{i,b} - Ny_{i,c} \le N$$

$$\ldots 1$$

$$\ldots + Nx_f + Nx_{\hat{f}} \le 2N$$

$$\ldots \theta_{\hat{f}} - Nx_f + Nx_{\hat{f}} \le N$$

If an agent is active

Each agent bids if it was not told it was winning i.e., whenever a CA to which it does not belong is selected

continuous variables

$$\ldots_{,pass} + \sum \theta_{i,b} = K_i$$

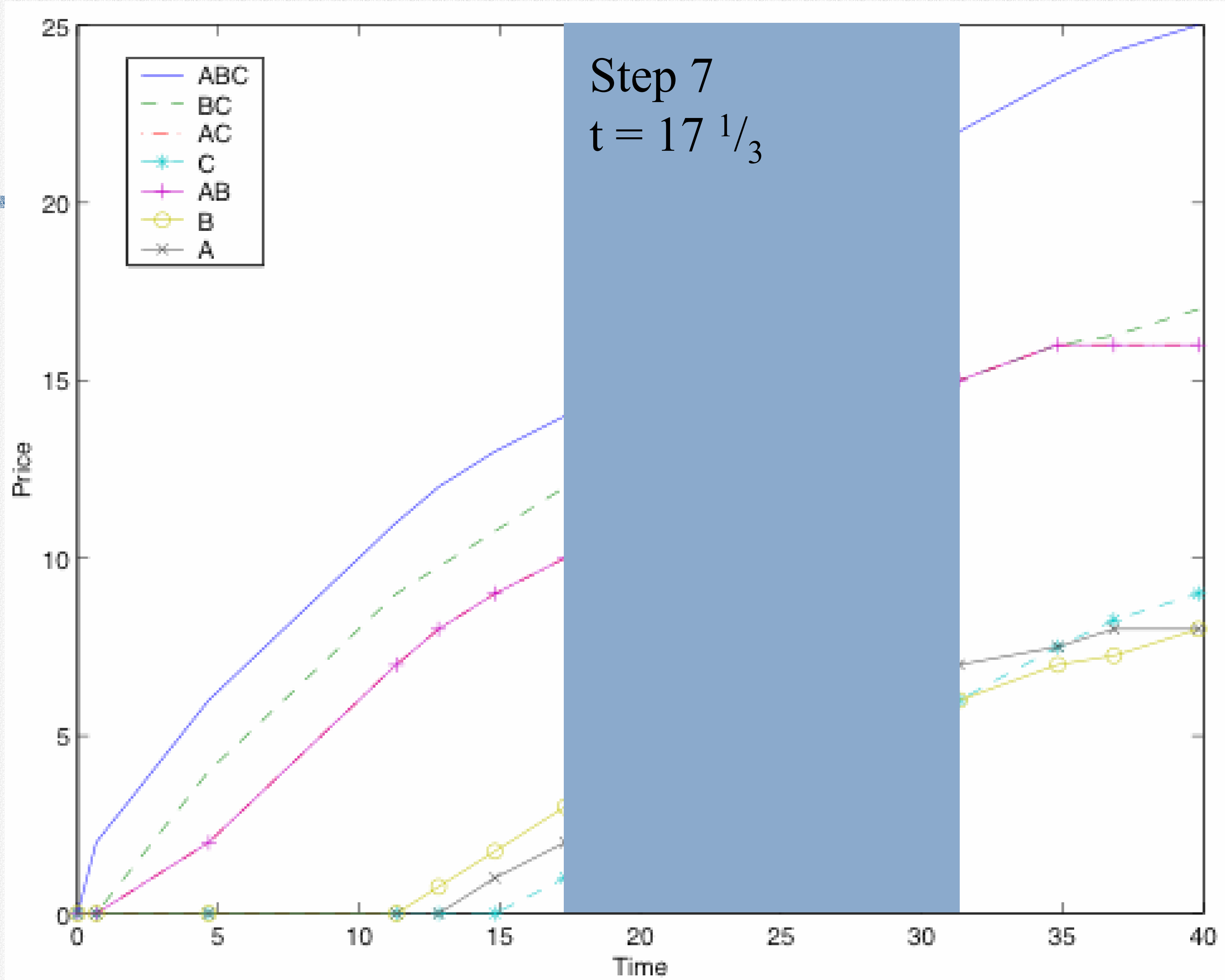
$$1 - \theta_{i,pass} = \sum (1 - G_{f,i}) \beta_f$$

$$\beta_f \le x_f \le \sum y_{i,f_i}$$

$$\theta_b \le y_{i,b} \le N\theta_b$$

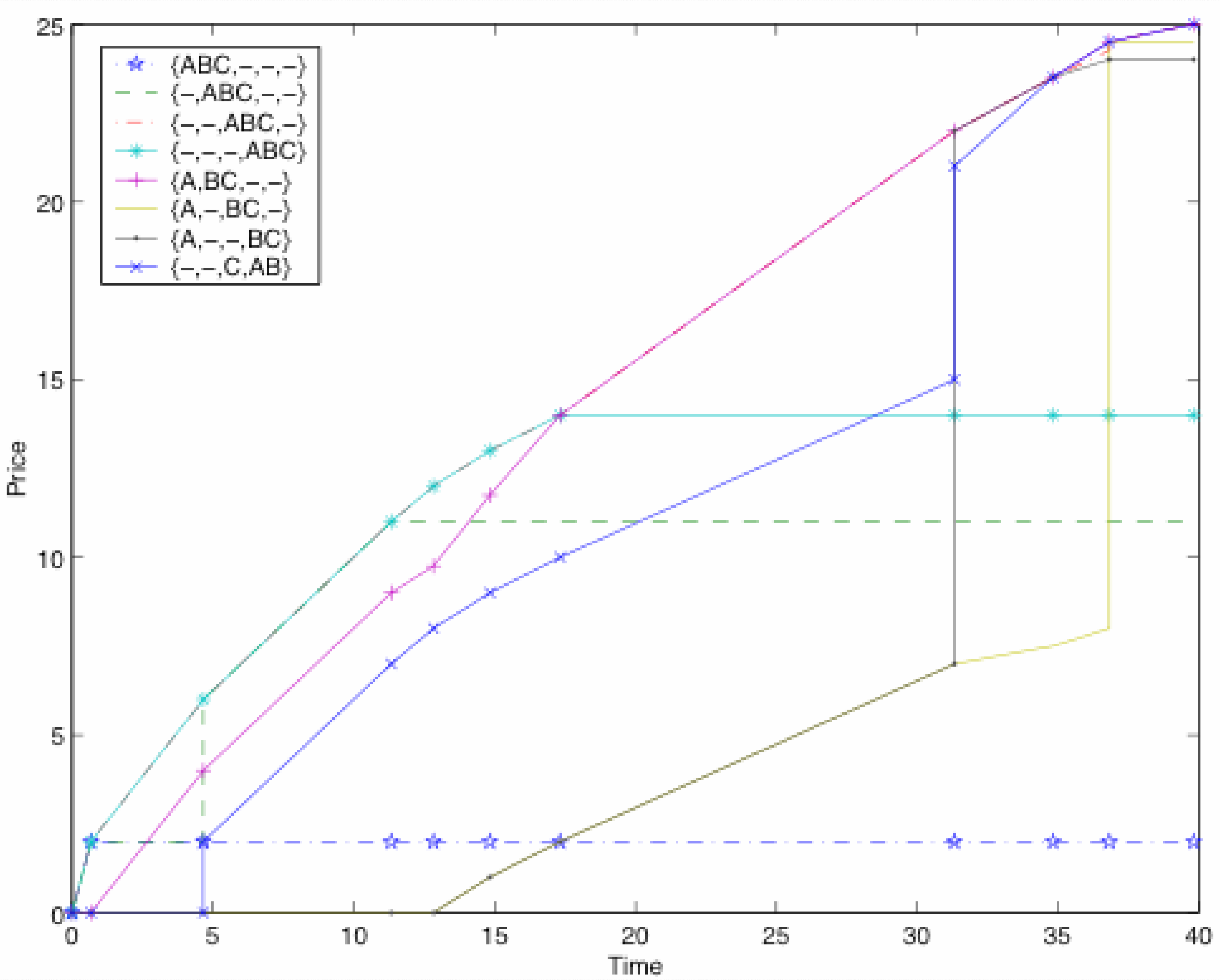
The Algorithm: Main Loop

- Solve the MILP to get
  - The demand set of each agent
  - The allocation of attention
  - The competitive allocations
- Compute the duration of the interval, or terminate
- Compute the prices at the end of the interval
- Jump to end of interval and repeat


Step 7
t = 17 1/3
ABC
BC
AC
C
AB
B
A
Price
Time
0
5
10
15
20
25
30
35
40
0
5
10
15
20
25

Step 7: The Allocation of Attention

	D_i	θ_A	θ_B	θ_{AB}	θ_C	θ_{AC}	θ_{BC}	θ_{ABC}	θ_{pass}
a_1	A, AB, AC	5/14				1/14			4/7
a_2	B, BC		3/14				3/14		4/7
a_3	ABC							4/7	3/7
a_4	AB, C, AC, ABC			5/14	5/14	4/14			
	slope	5/14	3/14	5/14	5/14	5/14	3/14	4/7	


{ABC,–,–,–}
{–,ABC,–,–}
{–,–,ABC,–}
{–,–,–,ABC}
{A,BC,–,–}
{A,–,BC,–}
{A,–,–,BC}
{–,–,C,AB}
Price
Time
0
5
10
15
20
25
30
35
40

Anecdotal Comparison

- Simulation:
  - With $\delta = .005$, took > 3000 iterations
  - Accuracy depends on δ
  - Depends on tie-breaking rules, ordering of bidders
- Price Trajectory Algorithm
  - 11 computations
  - Focused only on points at which the behavior changed
  - Exact computation of prices and allocation

Some Comments

- Does not require complete value statements
- The algorithm handles multiple value statements

Directions

- Current implementation in AMPL
- Working on a systematic comparison of performance
- Improve computation time
- Prove correspondence with simulation
- Apply framework to other iterative combinatorial auctions

Questions?