Combinatorial Auctions

Yeau Shoham

What are combinatorial auctions (CAs)

- Multiple goods are auctioned simultaneously
- Each bid may claim any combination of goods
- A typical combination: a bundle ("Bid $100 for the TV, VCR and couch")
- More complex combinations are possible

Motivation: complementarity and substitutability

- Complementary goods have a superadditive utility function:
  - \( v(a,b) > v(a) + v(b) \)
  - In the extreme, \( v(a,b) \) and but \( v(a) + v(b) \) – that's different from the function for different substitutes

Overview of Lecture

- What can you bid: The expressive power of different bidding languages
- What should you bid: A taste for the game theory of CAs
- Computational complexity of CAs
Bidding Language Requirements

A bid is a declaration of a valuation function; the bidding language must be:

- Expressive
  - Enough to represent all valuation functions
- Concise
- Natural
  - Easy for humans to understand
- Tractable
  - Easy for auctioneer algorithms to handle

Unstructured bidding is impractical

- Bidder sends his entire valuation function (over all possible allocations) to auctioneer.
- Problem: Exponentiation
- Bidder sends his valuation as a computer program
- Problem: requires programs to access by any auctioneer algorithm

Simple case: identical goods

- Additive valuation \( v(x) = c \)
- Single item valuation \( v(j)=c \) for all \( j \)
- General symmetric valuation:
  - If \( k \) items valued in \( p_i \)
  - \( v(x^k) = 2^k p_i \)
- Downward-closing valuation \( p_j \geq p_{j+1} \)

The alternative: structured bidding

- The basic building block: atomic bid (explicit AND)
  - Atomic \( (\text{takes a right, loses right}) \)
- More complex:
  - Spectrum \( [\text{frequency-x}] \) XOR \( [\text{frequency-y}] \)
  - Network link \( [\text{a-b,c-d}] \) XOR \( [\text{a-d,c-b}] \)
- Adding constraints:
  - PC configuration \( \{ \text{disk size \( > 10 GB \)}, \text{speed \( > 3 M h z \)} \} \)
  - Equality constraints \( \{ \text{chase \( with \)} \} \) of matching colors
  - Time constraints \( \{ \text{quick for 2 hours, for \( \text{free \ for \( 3 \) hours} \)} \}

What are the precise syntax and semantic?

Assumptions

- No externalities
- Free disposal
- Nothing-for-nothing

The general case (distinct goods)

- Atomic \( (\text{AND}) \) bid:
  - \( ([\text{TV, VCR,}\ X] \text{OR} [\text{guitar,}\ 1000]) \text{OR} ([\text{Xbox, TV}] \text{OR} [\text{TV, VCR}]) \)
  - Meaning: \( v(T) = 10 \) if \( T \neq 0 \) otherwise
- OR bid:
  - \( ([\text{TV, VCR}] \text{OR} [\text{guitar,} 1000]) \text{OR} ([\text{Xbox, TV}] \text{OR} [\text{TV, VCR}]) \)
  - Meaning: \( (v(T) \text{OR} v(S) = \max(v_{\text{TV}}, v_{\text{VCR}}, v_{\text{guitar}}, v_{\text{Xbox}}, v_{\text{TV, VCR}})) \times (\frac{1}{v(T)}) \)
  - Note: \( v([\text{TV, VCR}, X] = 1000); \text{not} \ 30 \times 2 \)
- XOR bid:
  - \( ([\text{TV, VCR}] \text{OR} [\text{book,} 20]) \text{OR} ([\text{TV, DVD}] \text{OR} [\text{Xbox, TV}] \text{OR} [\text{VCR, TV}]) \)
  - Meaning: \( (v(T) \text{OR} v(S) = \max(v_{\text{TV}}, v_{\text{VCR}}, v_{\text{book}}, v_{\text{TV, VCR}})) \times (\frac{1}{v(T)}) \)

CS10, Spring 2012  (c) Reuben 7

CS10, Spring 2012  (c) Reuben 8

CS10, Spring 2012  (c) Reuben 9

CS10, Spring 2012  (c) Reuben 10

CS10, Spring 2012  (c) Reuben 11

CS10, Spring 2012  (c) Reuben 12
Expressive Power and Conciseness

**Theorem:** OR bids can represent all valuations without substitutability.

**Theorem:** NOR bids can represent all valuations.

**Theorem:** Additive valuations can be represented linearly with OR bids, but only exponentially with XOR bids.

More Complex Languages

- OR-of-XORs
- XOR-of-ORs
- other boolean structures...

'Dummy' Goods

- \((a, 0) \text{ XOR } (b, 0) \Rightarrow (a \lor b, 0) \text{ OR } ((a, 0) \lor (b, 0), 10)\)
- a 'dummy' good
- The ideas: any decent CA will never grant that bids simultaneously

- With dummy goods, OR can represent any boolean function.

- How many dummy goods are needed?
  - In the worst case, exponentially many
  - Example: the majority function
  - OR-of-XORs \(s\), where \(s\) is the number of atomic bids in the input
  - XOR-of-ORs \(s^3\)

Tractability

- Bid interpretation: Given the bid and a set of goods, determine the valuation of the set.
- atomic, XOR bids integrated in polynomial (indeed, linear) time.
- All other bid formats: require exponential time.

Overview of Lecture

- What can you bid: The expressive power of different bidding languages
  - What should you bid: A taste for the game theory of CAs
  - Computational complexity of CAs

Two yardssticks for auction design

- Revenue maximization: The seller should extract the highest possible price
  - Efficiency: The buyer(s) with the highest valuation get the good(s)
  - The latter is usually achieved by ensuring "incentive compatibility" - bids are incented to bid their true value, and hence maximizing over those bids also ensures efficiency.

Is a CA efficient? Does it maximize revenue?
The Naïve CA is not incentive compatible

- Naïve CA: Given a set of bids on bundles, auctioneer finds a subset containing non-conflicting bids that maximizes revenue, and charges each winning bidder his bid.
- This is not incentive compatible, and thus not (economically) efficient.
- Example:
  - \( v_j = 50, \) \( v_j = -50, \) \( v_j(x,y) = 100 \)
  - \( v_j = 75, \) \( v_j = 0, \) \( v_j(x,y) = 75 \)
  - Bidder 1 has incentive to "lie" and claim
    - \( v_j = 70, v_j = 0, v_j(x,y) = 100 \)

Lessons from the single dimensional case

- 1st price sealed bid auction is not incentive compatible (neqilibrium, it pays to "shade" a bit off your true value)
- 2nd price sealed bid ("Vickrey") auction is incentive compatible
- Can we pull off the same trick here?

The Generalized Vickrey Auction (GVA)* is incentive compatible

- The Generalized Vickrey Auction charges each bidder the social cost.
- Example:
  - Red bids 30 for \([a] \), Green bids 19 for \([a,b] \), Blue bids 8 for \([b] \)
  - Naïve Green gets \([a,b] \) and pays 19
  - GVA Green gets \([a,b] \) and pays 18 (10 due to Red, 8 due to Blue)

* aka the VCG or VCG (Vicary, Clarke, Groves) mechanism

Formal definition of GVA

- Each bidder is given a utility function \( f(\cdot) \) possibly different from \( u_i(\cdot) \)
- The center calculates \( x^* \) which maximizes sum of \( f(\cdot) \)
- The center calculates \( \lambda_x \) which maximizes sum of \( f(\cdot) \) without \( i \)
- Agent \( i \) receives his share of \( x^* \) and also a payment of 
  \[ \sum_{j \neq i} f_j(x) - \sum_{j \neq i} f_j(\lambda_{x(i)}) \]
- Thus agent \( i \)'s utility is 
  \[ u_i(x^*) + \sum_{j \neq i} f_j(x) - \sum_{j \neq i} f_j(\lambda_{x(i)}) \]

What should agent \( i \) bid?

- Of the overall reward 
  \[ u_i(x^*) + \sum_{j \neq i} f_j(x) - \sum_{j \neq i} f_j(\lambda_{x(i)}) \]
- \( i \)'s bid impacts only 
  \[ u_i(x^*) + \sum_{j \neq i} f_j(x) \]
- but the auctioneer maximizes 
  \[ f_j(x^*) + \sum_{j \neq i} f_j(x) = \sum_j f_j(x) \]
- therefore \( i \) should make sure his function is identical to the auctioner's!

Other remarks about GVA

- Applies not only to auctions as we know them, but to general resource allocation problems
  - When "externalities" exist
  - E.g., with public goods
  - Cannot simultaneously guarantee
    - Participation
    - Incentive compatibility
    - Budget balance
    - Not collusion-proof
Overview of Lecture

- What can you bid: The expressive power of different bidding languages
- What should you bid: A taste for the game theory of CAs
- Computational complexity of CAs

What’s known about the problem?

- Known as the Set Packing Problem (SPP)
- It is NP-complete, meaning that effectively the only algorithms guaranteed to find the optimal solution will run exponentially long in the worst case
- Furthermore, you cannot even uniformly approximate the optimal solution \( K \leq \infty \) an algorithm that can guarantee that you always reach within a fixed fraction of \( K \), no matter how small the fraction, although you can get within \( 1/\sqrt{K} \) of \( K \), where \( K \) is the number of goods
- Nonetheless, progress has been made recently on algorithms optimised for this problem...

Approaches to taming the computational complexity of CAs

- Finding tractable special cases
- LP-relaxation of the IP problem
- Applying classic heuristic methods
- Applying incomplete heuristic methods
- How to use these algorithms? The need for a test suite
- Learning where the hard problems lie

SPP as an Integer Program

- \( n \) items – indexed by \( i \) (some may be dummy)
- \( m \) atomic bids: \( (S, p) \) (may be multiple sheets from some dataset)
- Goal: optimise social efficiency
  
  \[
  \text{Maximise } \sum_{i=1}^{n} x_i p_i \\
  \text{Subject to:} \\
  \sum_{i=1}^{n} x_i \leq 1 \quad \forall i \\
  x_i \in \{0, 1\} \quad \forall j
  \]

Linear Programming Relaxation of the IP

\[
\text{Maximise } \sum_{i=1}^{n} x_i p_i \\
\text{Subject to:} \\
\sum_{i=1}^{n} x_i \leq 1 \quad \forall i \\
\sum_{j=1}^{m} x_j \leq 1 \quad \forall i \\
x_i \in \{0, 1\} \quad \forall j
\]

- Good news: LP is easy
- Bad news: Will produce “fractional” allocations: \( x \) specifies what fraction of bid \( j \) is obtained.
- Pretty good news: If we’re lucky, the solution will be integer anyway
When do we get lucky?

- Tree saturated bundles; e.g., wine cases
- Contiguous single-dimensional goods; e.g., time intervals
- A general condition: Total Unimodular (TU) matrices
- Bundles of size at most 2

Tree-Structured Bundles

Example: Wine cases

- Directed algorithm: bottom-up maximization
- Compare the maximum among the value of the set of leaves and the sum of their children
- Q(t): value of the tree at time t

More General: Contiguous 1-Dimensional Goods

- Example: Blocks of time, contiguos lots
- Tree algorithm: recursive procedure
  - Consider the 1-dimensional good $a$, $b$, $c$, $d$, $e$, $f$, $g$
  - Which moves $1$ or $2$ units for all intervals is, $a$, $ab$, $abc$, $bc$, $bcd$, $abcd$, etc.
  - Now consider, say, $a$, $a$, $a$, $a$ for the optimal profits of all prefixes
  - Inductive step:
    - Assume you’ve found the optimal revenue for $a$, $ab$, $abc$, $bcd$, etc.
    - $g$ will either be used once, again $g$, sample $c$, $cd$, $bcd$, etc.; by induction, in each case you know how to maximize the revenue for the actual prefix
- Complexity: $O(n^3)$

Generalizing Both: Totally Unimodular (TU) Matrices

- Problem in matrix form:
  
  $\begin{array}{c|ccccc}
  \text{good} & 1 & 0 & 0 & 0 & 0 \\
  \hline
  100 & 1 & 0 & 0 & 0 & 0 \\
  011 & 0 & 1 & 0 & 0 & 0 \\
  001 & 0 & 0 & 1 & 0 & 0 \\
  010 & 0 & 0 & 0 & 1 & 0 \\
  000 & 0 & 0 & 0 & 0 & 1 \\
  \end{array}$

- $M$ is TU iff the determinants of each of its square submatrices is 1, 0, or $-1$
- In this case the solution to the LP is integer
- Complexity: $O(n^3)$
- Observation: will hold when you allow multiple units of each good, but still allow each bidder at most one unit of each good

State of the art regarding the general case

- In recent years, there have been an explosion of specialized search algorithms for CAs
- Complete methods guarantee optimal results, but not quick convergence. Most cases the algorithm scales to at least 200 goods and 10,000 bids.
- Incomplete greedy search methods sometimes perform an order of magnitude faster
- A major challenge: using the algorithms
  - Active learning, CATS (Hanne)
  - Using machine learning to find better and faster heuristics
required material on auctions, posted on web page, in addition this presentation and the one of April 25

Introduction to Multi-Agent Systems (draft)
Chapter 7: Mechanism Design
sections 7.3 and 7.4
by Y. Shoham (with T. Grainger)
(Only sections 7.2 and 7.4 are required; the rest are included just in case you’re curious)

---

**Some remaining issues on auctions**

- Two-sided markets
- Beyond zoology

---

**Double markets: m'th and m+1'st prices**

<table>
<thead>
<tr>
<th>BUY</th>
<th>SELL</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>32</td>
</tr>
<tr>
<td>47</td>
<td>33</td>
</tr>
<tr>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>34</td>
<td>45</td>
</tr>
<tr>
<td>34</td>
<td>54</td>
</tr>
</tbody>
</table>

---

**Double markets: the “Bid-Ask” Spread**

<table>
<thead>
<tr>
<th>BID</th>
<th>ASK</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>52</td>
</tr>
<tr>
<td>47</td>
<td>57</td>
</tr>
<tr>
<td>35</td>
<td>65</td>
</tr>
<tr>
<td>34</td>
<td>65</td>
</tr>
<tr>
<td>34</td>
<td>74</td>
</tr>
</tbody>
</table>

---

**What we’ve done so far: zoological categorization**

- single-dimensional auctions
  - open outcry
  - sealed bid
    - one-sided
    - two-sided
  - one-sided
  - two-sided

---

**A deeper look at what auctions really are**

*Definition: An auction is any negotiation mechanism that is:*
  - Mediated
  - Well-specified (e.g., according to English laws)
  - Market-based (determines an exchange in terms of standard currency)
Auctioneer activities

- Receive bids
- Disseminate information
- Arrange trades (clear market)

Auction (more generally: Trading) Dimensions

- Bidding rules
- Clearing policy
- Information revelation policy

Ramifications

- Software engineering
- Beyond auctions/bargaining, negotiations