Combinatorial Auctions

Yoav Shoham

What are combinatorial auctions (CAs)

- Multiple goods are auctioned simultaneously
- Each bid may claim any combination of goods
- A typical combination: a bundle ("I 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, $V(\{a,b\}) >> 0$ but $V(\{a\}) = V(\{b\}) = 0$
  - Example: different segments of a flight

- Substitutable goods have a subadditive utility function:
  - $V(\{a,b\}) < V(\{a\}) + V(\{b\})$
  - In the extreme, $V(\{a,b\}) = MAX[\, V(\{a\}) , V(\{b\}) \,]$
  - Examples: a United ticket and a Delta ticket

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

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

Unstructured bidding is impractical

- Bidder sends his valuation v as a vector of numbers to auctioneer.
  - Problem: Exponential size

- Bidder sends his valuation v as a computer program (applet) to auctioneer.
  - Problem: requires exponential access by any auctioneer algorithm

In practice bids have specific formats

- “Classic”:
  - (take-off right) AND (landing right)
  - (frequency A) XOR (frequency B)
- Online Computational resources:
  - Links: ((a--b) AND (b--c)) XOR ((a--d) AND (d--c))
  - (disk size > 10G) AND (speed >1M/sec)
- E-commerce:
  - chair AND sofa -- of matching colors
  - (machine A for 2 hours) AND (machine B for 1 hour)

Bidding Language Requirements

- Expressiveness
  - Must be expressive enough to represent every possible valuation.
  - Representation should not be too long
- Simplicity
  - Easy for humans to understand
  - Easy for auctioneer algorithms to handle

AND, OR, and XOR bids

- {left-sock, right-sock}:10
- {blue-shirt}:8 XOR {red-shirt}:7
- {stamp-A}:6 OR {stamp-B}:8

General OR bids and XOR bids

- {a,b}:7 OR {d,e}:8 OR {a,c}:4
  - {a}=0, {a, b}=7, {a, c}=4, {a, b, c}=7, {a, b, d, e}=15
  - Can only express valuations with no substitutabilities.
- {a,b}:7 XOR {d,e}:8 XOR {a,c}:4
  - {a}=0, {a, b}=7, {a, c}=4, {a, b, c}=7, {a, b, d, e}=8
  - Can express any valuation
  - Requires exponential size to represent

    {a}:1 OR {b}:1 OR … OR {z}:1

OR of XORs example

{couch}:7 XOR {chair}:5

OR

{TV, VCR}:8 XOR {Book}:3

Relative expressive power of different formats

- OR bids can represent valuations without substitutabilities
- XOR bids can represent all valuations
- Additive valuations can be represented linearly with OR bids, but only exponentially with XOR bids

The expressive power of ‘dummy’ (‘phantom’) goods

- Transform “$10 for a XOR (b and c)” into two bids: “$10 for a and x” and “$10 for b, c and x”; x is the dummy good.
  - The idea: any decent CA will never grant the two bids
- With dummy goods, OR can represent any function
- How many dummy goods are needed?
  - In the worst case, exponentially many
    - Example: the Majority valuation
  - OR-of-XORs: s, where s is the number of atomic bids in the input
  - XOR-of-ORs: s^2

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 yardsticks 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" – bidders are incented to bid their truth 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, find a subset containing non-conflicting bids that maximizes revenue, and charge each winning bidder his bid
- This is not incentive compatible, and thus not (economically) efficient
- Example:
  - v1(x,y)=100, v1(x)=v2(x)=0
  - v2(x,y)=0, v2(x)=v2(y)=75
  - Bidder 1 has incentive to “lie” and bid 76; if bidder 2 lies then bidder 1 has an incentive to lie even more

Lessons from the single dimensional case

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

The Generalized Vickrey Auction (GVA)* is incentive compatible

- The Generalized Vickrey Auction charges each bidder their social cost

- Example:
  - Red bids 10 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 Vickrey-Clarke-Groves (VCG) mechanism

Formal definition of GVA

- Each *i* reports a utility function $r_i(\cdot)$ possibly different from $u_i(\cdot)$
- The center calculates (x^*) which maximizes sum of r_is
- The center calculates $(\hat{x}_{-i})$ which maximizes sum of r_is without *i*
- Agent *i* receives (x_i^*) and also a payment of

$$\sum_{j\neq i} r_j(x^*) - \sum_{j\neq i} r_j(\hat{x}_{\sim i})$$

- Thus agent *i*'s utility is

$$u_i(x^*) + \sum_{j\neq i} r_j(x^*) - \sum_{j\neq i} r_j(\hat{x}_{\sim i})$$

What should agent *i* bid?

Of the overall reward $u_i(x^*) + \sum_{j \neq i} r_j(x^*) - \sum_{j \neq i} r_j(\hat{x}_{\sim i})$

i's bid impacts only $u_i(x^*) + \sum_{j \neq i} r_j(x^*)$

the auctioneer maximizes $r_i(x^*) + \sum_{j \neq i} r_j(x^*) = \sum_j r_j(x^*)$

therefore *i* should make sure his function is identical to the auctioneer's!

Other remarks about GVA

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

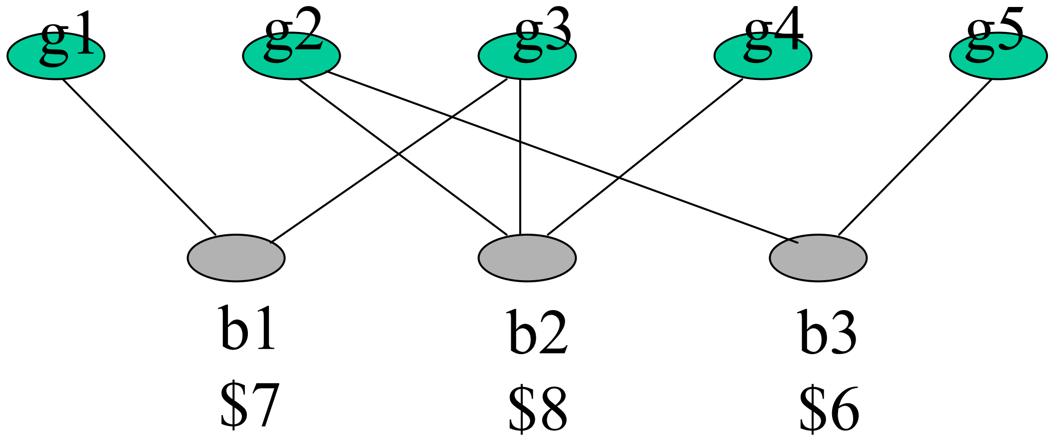
Overview of Lecture

- 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

The optimization problem of CAs

- “Given a set of bids on bundles, find a subset containing non-conflicting bids that maximizes revenue”
- Performed once by the naïve method, n+1 times by GVA
- Requires exponential time in the number of goods and bids (assuming they are polynomially related)



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 (there isn't an algorithm that can guarantee that you always reach within a fixed fraction of it, no matter how small the fraction, although you can get within $1/\sqrt{k}$ of it, where K is the number of goods)
- Nonetheless, progress has been made recently on algorithms optimized for this problem…

Approaches to taming the computational complexity of CAs

- Finding tractable special cases
- LP-relaxation of the IP problem
- Applying complete heuristic methods
- Applying incomplete heuristic methods
- How to test these algorithms? The need for a test suite

SPP as an Integer Program

- n items -- indexed by i
  (some may be phantom)
- m atomic bids: (S_j, p_j)

(maybe multiple ones from same bidder)

- Goal: optimize social efficiency
- Problem: IP is hard

$$Maximize \sum_{j=1}^{m} x_j p_j$$

$$Subject \quad to:$$

$$\sum_{i \in S_j} x_j \leq 1 \quad \forall i$$

$$x_j \in \{0,1\} \quad \forall j$$

Linear Programming Relaxation of the IP

- Will produce “fractional” allocations: x_j specifies what fraction of bid j is obtained.
- LP is easy
- If we are lucky, the solution will be 0,1

$$Maximize \sum_{j=1}^{m} x_j p_j$$

$$Subject \quad to:$$

$$\sum_{i \in S_j} x_j \leq 1 \quad \forall i$$

$$x_j \geq 0 \quad \forall j$$

In matrix form

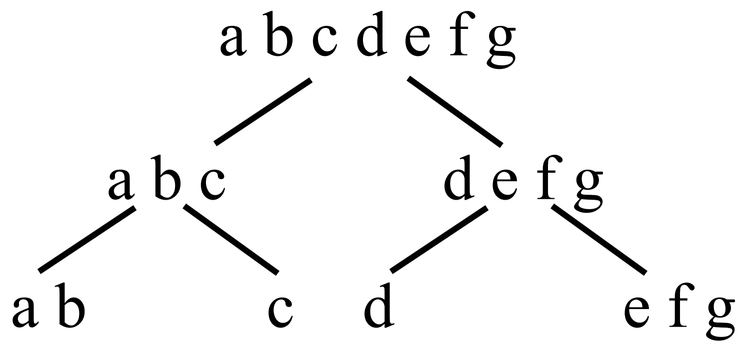
$$\max \sum_{S \subset M} b^*(S) x_s$$

$$s.t. \sum_{S : i \in S} x_s \leq 1 \forall i \in M$$

$$x_s = 0,1 \forall S \subset M$$

When do we get lucky?

- Tree structured bundles:



- Continguous single-dimensional goods ("consecutive ones"); e.g., time intervals
- Bundles of size at most 2 (quadratic complexity)
- A general condition: Total Unimodular matrices

State of the art

- Recent years have seen an explosion of specialized search algorithms for CAs
- Complete methods guarantee optimal results, but not quick convergence. On test cases the algorithms scale to xx goods and xxxxxx bids.
- Incomplete, greedy-search methods sometimes perform an order of maginitude faster
- Very recent results on the multi-unit case
- CPLEX 7.0 holding its own…
- A major challenge: testing the algorithms (CATS)

Other handouts posted on web page

- *Combinatorial Auctions: A Survey*, by de Vries and Vohra
  - Only pp. 1-14 (thru 2.3.1) required; rest optional
- *Mechanism Design for Computerized Agents*, Varian
- *Elements of Auction Theory*, Shoham
  - Optional; not required for the course