

CPS 590.4 Auctions & Combinatorial Auctions

Vincent Conitzer
conitzer@cs.duke.edu

A few different 1-item auction mechanisms

- **English** auction:

- Each bid must be higher than previous bid
- Last bidder wins, pays last bid

- **Japanese** auction:

- Price rises, bidders drop out when price is too high
- Last bidder wins at price of last dropout

- **Dutch** auction:

- Price drops until someone takes the item at that price

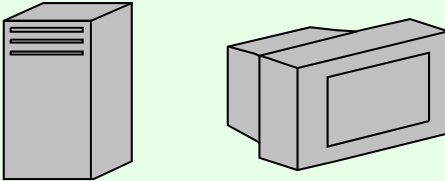

- **Sealed-bid** auctions (direct-revelation mechanisms):

- Each bidder submits a bid in an envelope
- Auctioneer opens the envelopes, highest bid wins


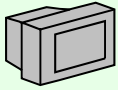

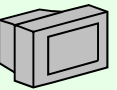
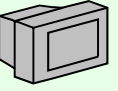
- **First-price** sealed-bid auction: winner pays own bid

- **Second-price** sealed bid (or **Vickrey**) auction: winner pays second-highest bid

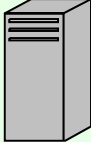
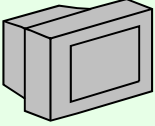

Complementarity and substitutability

- How valuable one item is to a bidder may depend on whether the bidder possesses another item
- Items a and b are **complementary** if $v(\{a, b\}) > v(\{a\}) + v(\{b\})$
- E.g. A server tower and a microwave oven.
- Items a and b are **substitutes** if $v(\{a, b\}) < v(\{a\}) + v(\{b\})$
- E.g. A laptop and a server tower.

Inefficiency of **sequential** auctions

- Suppose your valuation function is $v(\text{server}) = \$200$, $v(\text{printer}) = \$100$, $v(\text{server} \text{ printer}) = \500
- Now suppose that there are two (say, Vickrey) auctions, the first one for  and the second one for 
- What should you bid in the first auction (for )?
- If you bid \$200, you may lose to a bidder who bids \$250, only to find out that you could have won  for \$200
- If you bid anything higher, you may pay more than \$200, only to find out that  sells for \$1000
- Sequential (and **parallel**) auctions are **inefficient**

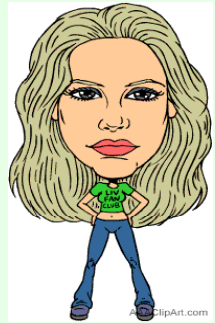
Combinatorial auctions

Simultaneously for sale:  ,  , 



bid 1

$$v(\text{server rack} \text{ cabinet}) = \$500$$



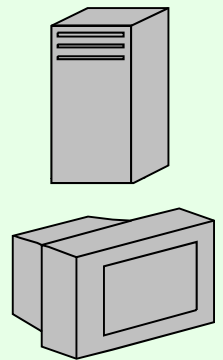
bid 2

$$v(\text{laptop} \text{ cabinet}) = \$700$$



bid 3

$$v(\text{laptop}) = \$300$$



used in truckload transportation, industrial procurement, radio spectrum allocation, ...

The winner determination problem (WDP)

- Choose a subset A (the accepted bids) of the bids B ,
- to maximize $\sum_{b \text{ in } A} v_b$,
- under the constraint that every item occurs at most once in A
 - This is assuming **free disposal**, i.e., not everything needs to be allocated

WDP example

- Items A, B, C, D, E
- Bids:
- ($\{A, C, D\}$, 7)
- ($\{B, E\}$, 7)
- ($\{C\}$, 3)
- ($\{A, B, C, E\}$, 9)
- ($\{D\}$, 4)
- ($\{A, B, C\}$, 5)
- ($\{B, D\}$, 5)
- *What's an optimal solution?*
- *How can we prove it is optimal?*

Price-based argument for optimality

- Items A, B, C, D, E
- Bids:
 - ($\{A, C, D\}$, 7)
 - ($\{B, E\}$, 7)
 - ($\{C\}$, 3)
 - ($\{A, B, C, E\}$, 9)
 - ($\{D\}$, 4)
 - ($\{A, B, C\}$, 5)
 - ($\{B, D\}$, 5)
- Suppose we create the following “prices” for the items:
 - $p(A) = 0$, $p(B) = 7$,
 $p(C) = 3$, $p(D) = 4$,
 $p(E) = 0$
 - Every bid bids at most the sum of the prices of its items, so we can't expect to get more than 14.

Price-based argument does not always give matching upper bound

- Clearly can get at most 2
- Items A, B, C
- Bids:
 - $(\{A, B\}, 2)$
 - $(\{B, C\}, 2)$
 - $(\{A, C\}, 2)$
- If we want to set prices that sum to 2, there must exist two items whose prices sum to < 2
- But then there is a bid on those two items of value 2
 - (Can set prices that sum to 3, so that's an upper bound)

Should not be surprising, since it's an NP-hard problem and we don't expect short proofs for negative answers to NP-hard problems (we don't expect $NP = coNP$)

An integer program formulation

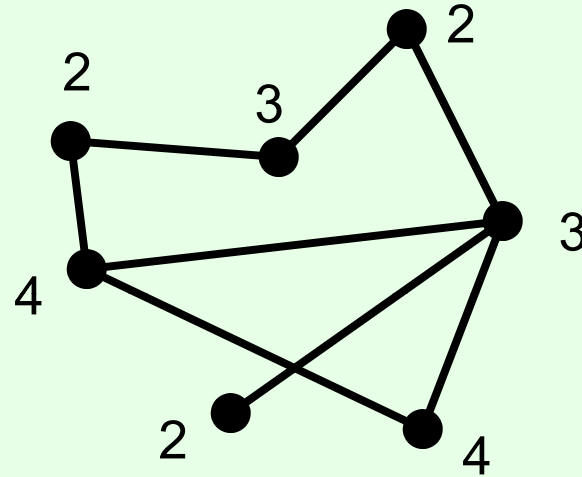
- x_b equals 1 if bid b is accepted, 0 if it is not
- maximize $\sum_b v_b x_b$
- subject to
 - for each item j , $\sum_{b: j \text{ in } b} x_b \leq 1$
- If each x_b can take any value in $[0, 1]$, we say that bids can be **partially accepted**
- In this case, this is a **linear** program that can be solved in polynomial time
- This requires that
 - each item can be divided into fractions
 - if a bidder gets a fraction f of **each** of the items in his bundle, then this is worth the same fraction f of his value v_b for the bundle

Price-based argument **does** always work for partially acceptable bids

- Items A, B, C
- Bids:
 - $(\{A, B\}, 2)$
 - $(\{B, C\}, 2)$
 - $(\{A, C\}, 2)$
- Now can get 3, by accepting half of each bid
- Put a price of 1 on each item

General proof that with partially acceptable bids, prices always exist to give a matching upper bound is based on linear programming duality

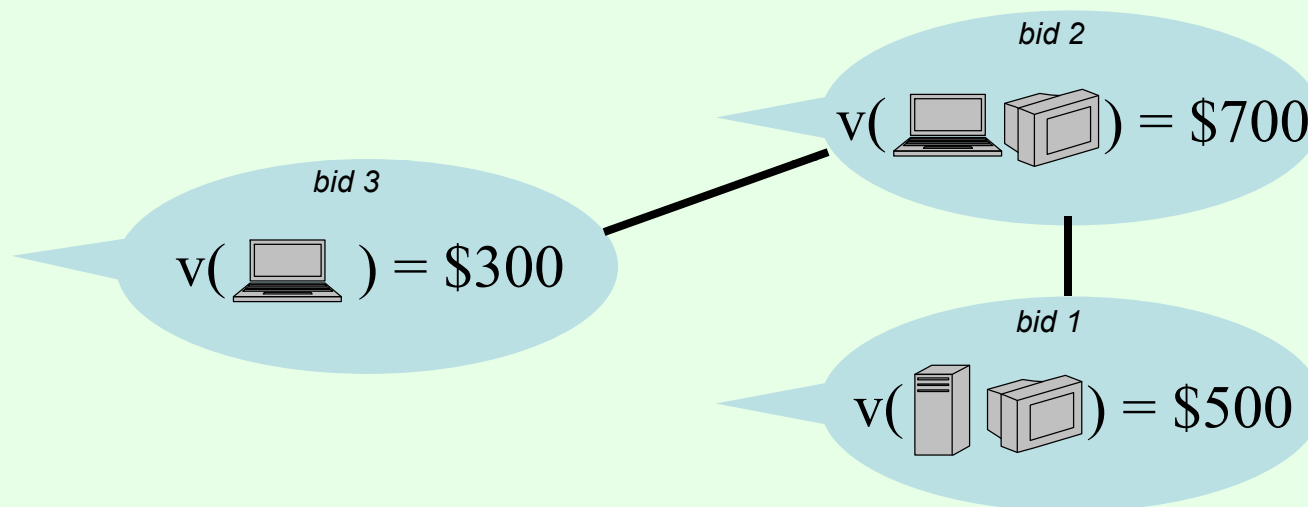
Weighted independent set



- Choose subset of the vertices with maximum total weight,
- Constraint: no two vertices can have an edge between them
- NP-hard (generalizes regular independent set)

The winner determination problem as a weighted independent set problem

- Each bid is a vertex
- Draw an edge between two vertices if they share an item



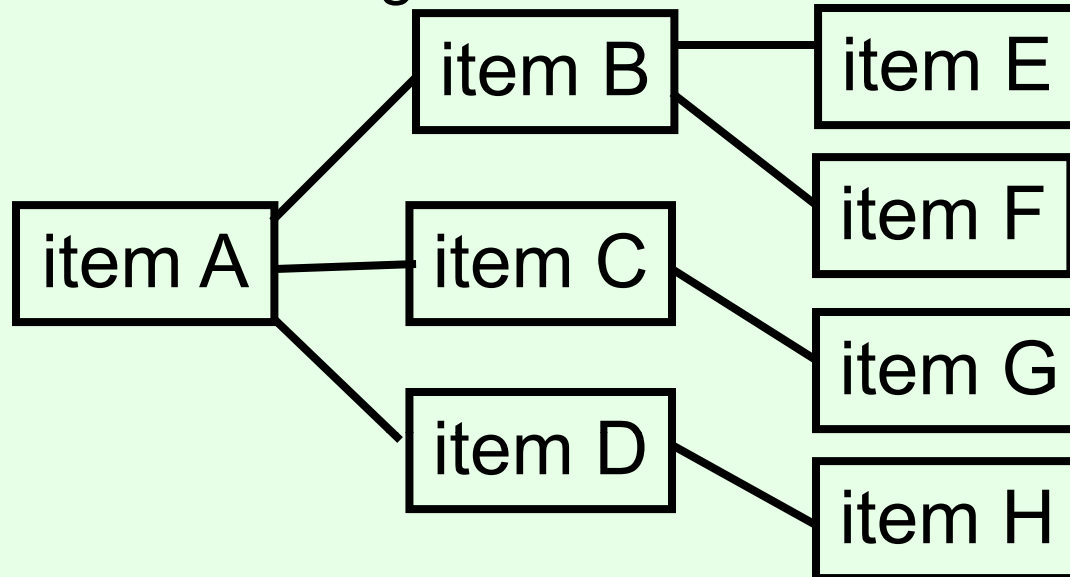
- Optimal allocation = maximum weight independent set
- Can model any weighted independent set instance as a CA winner determination problem (1 item per edge (or clique))
- Weighted independent set is NP-hard, even to solve approximately [Håstad 96] - hence, so is WDP
 - [Sandholm 02] noted that this inapproximability applies to the WDP

Dynamic programming approach to WDP [Rothkopf et al. 98]

- For every subset S of I , compute $w(S)$ = the maximum total value that can be obtained when allocating only items in S
- Then, $w(S) = \max \{ \max_i v_i(S), \max_{S': S' \text{ is a subset of } S, \text{ and there exists a bid on } S'} w(S') + w(S \setminus S') \}$
- Requires exponential time

Bids on connected sets of items in a tree

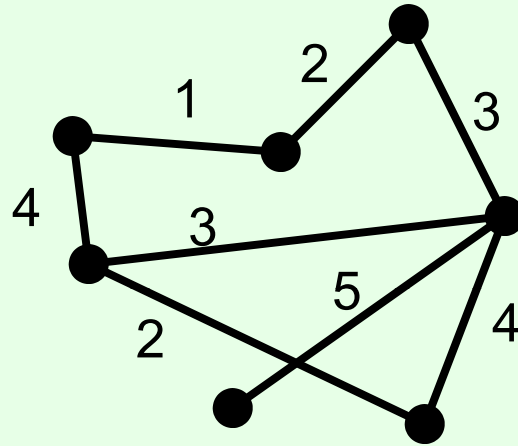
- Suppose items are organized in a tree



- Suppose each bid is on a connected set of items
 - E.g. {A, B, C, G}, but not {A, B, G}
- Then the WDP can be solved in polynomial time (using dynamic programming) [Sandholm & Suri 03]
- Tree does not need to be given: can be **constructed** from the bids in polynomial time if it exists [Conitzer, Derryberry, Sandholm 04]
- More generally, WDP can also be solved in polynomial time for graphs of **bounded treewidth** [Conitzer, Derryberry, Sandholm 04]
 - Even further generalization given by [Gottlob, Greco 07]

Maximum weighted matching

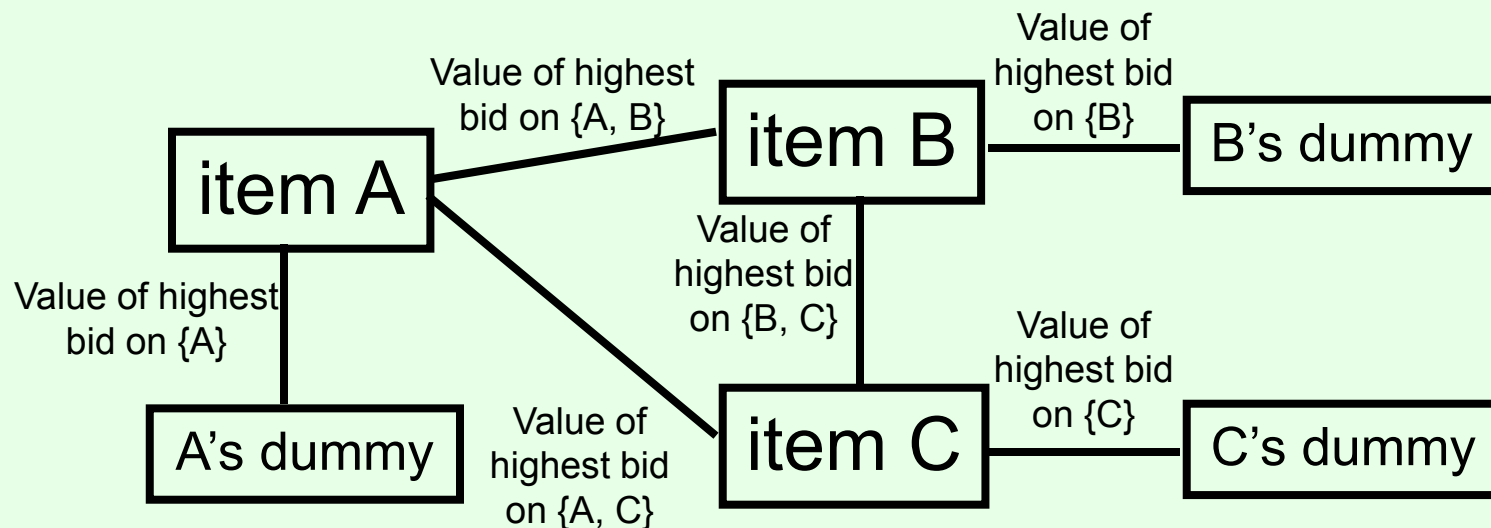
(not necessarily on bipartite graphs)



- Choose subset of the edges with maximum total weight,
- Constraint: no two edges can share a vertex
- Still solvable in polynomial time

Bids with few items [Rothkopf et al. 98]

- If each bid is on a bundle of at most **two** items,
- then the winner determination problem can be solved in polynomial time as a maximum weighted matching problem
 - 3-item example:



- If each bid is on a bundle of **three** items, then the winner determination problem is NP-hard again

Variants [Sandholm et al. 2002]: combinatorial reverse auction

- In a **combinatorial reverse auction (CRA)**, the auctioneer seeks to **buy** a set of items, and bidders have values for the different bundles that they may **sell** the auctioneer
 - minimize $\sum_b v_b x_b$
 - subject to
 - for each item j , $\sum_{b: j \text{ in } b} x_b \geq 1$

WDP example (as CRA)

- Items A, B, C, D, E
- Bids:
- ($\{A, C, D\}, 7$)
- ($\{B, E\}, 7$)
- ($\{C\}, 3$)
- ($\{A, B, C, E\}, 9$)
- ($\{D\}, 4$)
- ($\{A, B, C\}, 5$)
- ($\{B, D\}, 5$)

Variants:

multi-unit CAs/CRAs

- **Multi-unit** variants of CAs and CRAs: multiple units of the same item are for sale/to be bought, bidders can bid for multiple units
- Let q_{bj} be number of units of item j in bid b , q_j total number of units of j available/demanded
 - maximize $\sum_b v_b x_b$
 - subject to
 - for each item j , $\sum_b q_{bj} x_b \leq q_j$
 - minimize $\sum_b v_b x_b$
 - subject to
 - for each item j , $\sum_b q_{bj} x_b \geq q_j$

Multi-unit WDP example (as CA/CRA)

- Items: 3A, 2B, 4C, 1D, 3E
- Bids:
- ($\{1A, 1C, 1D\}$, 7)
- ($\{2B, 1E\}$, 7)
- ($\{2C\}$, 3)
- ($\{2A, 1B, 2C, 2E\}$, 9)
- ($\{2D\}$, 4)
- ($\{3A, 1B, 2C\}$, 5)
- ($\{2B, 2D\}$, 5)

Variants: (multi-unit) combinatorial exchanges

- **Combinatorial exchange (CE)**: bidders can **simultaneously** be buyers and sellers
 - Example bid: “If I receive 3 units of A and -5 units of B (i.e., I have to give up 5 units of B), that is worth \$100 to me.”
- maximize $\sum_b v_b x_b$
- subject to
 - for each item j , $\sum_b q_{b,j} x_b \leq 0$

CE WDP example

- Bids:
- $(\{-1A, -1C, -1D\}, -7)$
- $(\{2B, 1E\}, 7)$
- $(\{2C\}, 3)$
- $(\{-2A, 1B, 2C, -2E\}, 9)$
- $(\{-2D\}, -4)$
- $(\{3A, -1B, -2C\}, 5)$
- $(\{-2B, 2D\}, 0)$

Variants: no free disposal

- Change all inequalities to equalities

(back to 1-unit CAs) Expressing valuation functions using bundle bids

- A bidder is **single-minded** if she only wants to win one particular bundle
 - Usually not the case
- But: one bidder may submit multiple bundle bids
- Consider again valuation function $v(\text{server}) = \$200$, $v(\text{printer}) = \$100$, $v(\text{server}, \text{printer}) = \500
- What bundle bids should one place?
- What about: $v(\text{laptop}) = \$300$, $v(\text{server}) = \$200$, $v(\text{server}, \text{laptop}) = \400 ?

Alternative approach: report entire valuation function

- I.e., every bidder i reports $v_i(S)$ for every subset S of I (the items)
- Winner determination problem:
- Allocate a subset S_i of I to each bidder i to maximize $\sum_i v_i(S_i)$ (under the constraint that for $i \neq j$, $S_i \cap S_j = \emptyset$)
 - This is assuming free disposal, i.e., not everything needs to be allocated

Exponentially many bundles

- In general, in a combinatorial auction with set of items I ($|I| = m$) for sale, a bidder could have a different valuation for every subset S of I
 - Implicit assumption: **no externalities** (bidder does not care what the other bidders win)
- Must a bidder communicate 2^m values?
 - Impractical
 - Also difficult for the bidder to evaluate **every** bundle
- Could require $v_i(\emptyset) = 0$
 - Does not help much
- Could require: if S is a superset of S' , $v(S) \geq v(S')$ (**free disposal**)
 - Does not help in terms of number of values

Bidding languages

- **Bidding language** = a language for expressing valuation functions
- A good bidding language allows bidders to **concisely** express **natural** valuation functions
- Example: the **OR** bidding language [Rothkopf et al. 98, DeMartini et al. 99]
- Bundle-value pairs are ORed together, auctioneer may accept any number of these pairs (assuming no overlap in items)
- E.g. $(\{a\}, 3)$ OR $(\{b, c\}, 4)$ OR $(\{c, d\}, 4)$ implies
 - A value of 3 for $\{a\}$
 - A value of 4 for $\{b, c, d\}$
 - A value of 7 for $\{a, b, c\}$
- Can we express the valuation function $v(\{a, b\}) = v(\{a\}) = v(\{b\}) = 1$ using the OR bidding language?
- OR language is good for expressing complementarity, bad for expressing substitutability

XORs

- If we use XOR instead of OR, that means that only **one** of the bundle-value pairs can be accepted
- Can express **any** valuation function (simply XOR together all bundles)
- E.g. $(\{a\}, 3) \text{ XOR } (\{b, c\}, 4) \text{ XOR } (\{c, d\}, 4)$ implies
 - A value of 3 for $\{a\}$
 - A value of 4 for $\{b, c, d\}$
 - A value of 4 for $\{a, b, c\}$
- Sometimes not very concise
- E.g. suppose that for any S , $v(S) = \sum_{s \text{ in } S} v(\{s\})$
 - How can this be expressed in the OR language?
 - What about the XOR language?
- Can also combine ORs and XORs to get benefits of both [Nisan 00, Sandholm 02]
- E.g. $((\{a\}, 3) \text{ XOR } (\{b, c\}, 4)) \text{ OR } (\{c, d\}, 4)$ implies
 - A value of 4 for $\{a, b, c\}$
 - A value of 4 for $\{b, c, d\}$
 - A value of 7 for $\{a, c, d\}$

WDP and bidding languages

- **Single-minded bidders** bid on only one bundle
 - Valuation is v for any subset including that bundle, 0 otherwise
- If we can solve the WDP for single-minded bidders, we can also solve it for the OR language
 - Simply pretend that each bundle-value pair comes from a different bidder
- We can even use the same algorithm when XORs are added, using the following trick:
 - For bundle-value pairs that are XORed together, add a **dummy item** to them [Fujishima et al 99, Nisan 00]
 - E.g. $(\{a\}, 3) \text{ XOR } (\{b, c\}, 4)$ becomes $(\{a, \text{dummy}_1\}, 3) \text{ OR } (\{b, c, \text{dummy}_1\}, 4)$
- So, we can focus on single-minded bids