Computational Social Choice: Spring 2009

Ulle Endriss
Institute for Logic, Language and Computation
University of Amsterdam

Plan for Today

Allocating goods to agents is a typical example for collective decision making. Auctions are standardised methods for doing this.

- Overview: auction protocols for allocating a *single item*
- Introduction to *combinatorial auctions* as mechanisms for deciding on the allocation of *sets of items*
- Analysis of the *winner determination problem* (which bidder should obtain which items?) of combinatorial auctions in detail: *computational complexity* and *algorithms*

Basic Auction Theory

General setting for "simple" auctions:

- one seller (the *auctioneer*)
- many *buyers*
- one single item to be sold, e.g.
  - a house to live in (*private value auction*)
  - a house that you may sell on (*correlated value auction*)

There are many different *auction mechanisms* or *protocols*, even for simple auctions ...

English Auctions

- Protocol: auctioneer starts with the *reservation price*; in each round each agent can propose a higher bid; final bid wins
- Used to auction paintings, antiques, etc.
- Dominant strategy (for private value auctions): bid a little bit more in each round, until you win or reach your own valuation
- Counterspeculation (how do others value the good on auction?) is not necessary.
- *Winner's curse* (in correlated value auctions): if you win but have been uncertain about the true value of the good, should you actually be happy?

Dutch Auctions

- Protocol: the auctioneer starts at a very high price and lowers it a little bit in each round; the first bidder to accept wins
- Used at the flower wholesale markets in Amsterdam.
- Intuitive strategy: wait for a little bit after your true valuation has been called and hope no one else gets in there before you (no general dominant strategy)
- Also suffers from the winner's curse.

First-price Sealed-bid (FPSB) Auctions

- Protocol: one round; sealed bid; highest bid wins (for simplicity, we assume no two agents make the same bid)
- Used for public building contracts etc.
- Problem: the difference between the highest and second highest bid is "wasted money" (the winner could have offered less).
- Intuitive strategy: bid a little bit less than your true valuation (no general dominant strategy)

Vickrey Auctions

- Proposed by William Vickrey in 1961 (Nobel Prize in Economic Sciences in 1996)
- Protocol: one round; sealed bid; highest bid wins, but the winner pays the price of the *second highest* bid
- Dominant strategy: bid your true valuation
  - if you bid more, you risk paying too much
  - if you bid less, you lower your chances of winning while still having to pay the same price in case you do win
- Problem: counterintuitive (problematic for humans)

W. Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *Journal of Finance*, 16(1):8–37, 1961.

Revenue for the Auctioneer

- Which protocol is best for the auctioneer?
- Revenue-equivalence Theorem (Vickrey, 1961):

  All four protocols give the same expected revenue for private value auctions amongst risk-neutral bidders with valuations independently drawn from a uniform distribution.

- Intuition: revenue ≈ second highest valuation:
  - Vickrey: clear ✓
  - English: bidding stops just after second highest valuation ✓
  - Dutch/FPSB: because of the uniform value distribution, top bid ≈ second highest valuation ✓
- But: this applies only to an artificial and rather idealised situation; in reality there are many exceptions.

Complements and Substitutes

The value an agent assigns to a *bundle* of goods may relate to the value it assigns to the individual goods in a variety of ways ...

- *Complements:* The value assigned to a set is *greater* than the sum of the values assigns to its elements.

  A standard example for complements would be a pair of shoes (a left shoe and a right shoe).

- *Substitutes:* The value assigned to a set is *lower* than the sum of the values assigned to its elements.

  A standard example for substitutes would be a ticket to the theatre and another one to a football match for the same night.

In such cases an auction mechanism allocating one item at a time is problematic as the best bidding strategy in one auction may depend on whether the agent can expect to win certain future auctions.

Combinatorial Auction Protocol

- Setting: one seller (*auctioneer*) and several potential buyers (*bidders*); many *goods* to be sold
- Bidding: the bidders *bid* by submitting their valuations to the auctioneer (not necessarily truthfully)
- Clearing: the auctioneer announces a number of *winning bids*

  The winning bids determine which bidder obtains which good, and how much each bidder has to pay. No good may be allocated to more than one bidder.

Bidding Languages

- As there are $2^n - 1$ non-empty bundles for n goods, submitting complete valuations may not be feasible.
- So we need a compact preference representation language (OR/XOR, weighted formulas, k-additive form, ...).
- Today: (mostly) *OR language*, i.e., assume each bidder submits a number of atomic bids (B_i, p_i), specifying the price p_i the bidder is prepared to pay for a particular bundle B_i.
- In general, we may think of the bidding language as determining a *conflict graph:* atomic bids are vertices and edges connect bids that cannot be accepted together.

The Winner Determination Problem

The *winner determination problem* (WDP) is the problem of finding a set of winning atomic bids

(1) that is feasible (e.g., no item allocated twice) and

(2) that will maximise the sum of the prices offered.

The sum of prices can be given different interpretations:

(1) If the simple pricing rule is used where bidders pay what they offered, then it is the revenue of the auctioneer.

(2) If the prices offered are interpreted as individual utilities, then it is the utilitarian social welfare of the selected allocation.

Complexity of Winner Determination

The decision problem underlying the WDP is NP-complete:

Theorem 1 *Let $K \in \mathbb{Z}$. The problem of checking whether there exists a solution to a given combinatorial auction instance generating a revenue exceeding K is NP-complete.*

This has first been stated by Rothkopf *et al.* (1998).

We have already seen a proof for this in the lecture on MARA: the problem is equivalent to Welfare Optimisation. Recall that proving NP-membership was easy and that NP-hardness followed from a reduction from Set Packing.

M.H. Rothkopf, A. Pekeč, and R.M. Harstad. Computationally Manageable Combinational Auctions. *Management Science*, 44(8):1131–1147, 1998.

Intractable Special Cases

There are various results that show that seemingly severe restrictions of the WDP *remain NP-hard*. For instance:

> *Winner determination remains NP-hard if each bidder only submits a single atomic bid and assigns it a price of 1.*

This immediately follows from the specific reduction from Set Packing that we have seen in an earlier lecture.

D. Lehmann, R. Müller, and T. Sandholm. The Winner Determination Problem. In P. Cramton *et al.* (eds.), *Combinatorial Auctions*, MIT Press, 2006.

Tractable Special Cases

Another line of research has tried to identify special cases for which the WDP becomes *tractable*. Such cases are characterised by specific structural properties of the valuations that bidders report.

Here is an example:

Theorem 2 (Rothkopf *et al.*, 1998) *If the conflict graph is a tree, then the WDP can be solved in polynomial time.*

Proof sketch: Start from the leaves of the tree, going up. Accept a bid iff it has a higher price than the best combination you can get from its offspring. ✓

M.H. Rothkopf, A. Pekeč, and R.M. Harstad. Computationally Manageable Combinational Auctions. *Management Science*, 44(8):1131–1147, 1998.

Solving the Winner Determination Problem

We have seen that the WDP is intractable (NP-complete) in its general form. Nevertheless, sophisticated algorithms often manage to solve even large CA instances in practice.

There are two types of approaches to *optimal* winner determination in the *general* case:

- Use powerful general-purpose *mathematical programming* (integer programming) software (next slide)
- Develop search algorithms specifically for winner determination, combining general *AI search techniques* and domain-specific heuristics (rest of this lecture)

Other options include developing special-purpose algorithms for *tractable subclasses* (as discussed) and *approximation algorithms* for the general case (which we won't discuss here).

Integer Programming Approach

Suppose bidders submit n bids as bundle/price pairs (B_i, p_i) with the implicit understanding that the auctioneer may accept any combination of non-conflicting bids and charge the sum of the associated prices (i.e., we are using the OR bidding language).

Introduce a decision variable $x_i \in \{0, 1\}$ for each bid (B_i, p_i).

The WDP becomes the following Integer Programming problem:

- Maximise $\sum_{i=1}^{n} p_i \cdot x_i$ subject to $\sum_{i \in Bids(g)} x_i \leq 1$ for all goods g, where $Bids(g) = \{i \in [1, n] \mid g \in B_i\}$

Highly optimised software packages for mathematical programming (such as CPLEX/ILOG) can often solve such problems efficiently.

Search for an Optimal Solution

Next we are going to see how to customise well-known search techniques developed in AI so as to solve the WDP.

This part of the lecture will largely follow the survey article by Sandholm (2006).

T. Sandholm. Optimal Winner Determination Algorithms. In P. Cramton *et al.* (eds.), *Combinatorial Auctions*, MIT Press, 2006.

Search Techniques in AI

A generic approach to search uses the *state-space representation:*

- Represent the problem as a set of *states* and define *moves* between states. Given an initial state, this defines a *search tree*.
- The *goal states* are states that correspond to valid solutions.
- Each move is associated with a *cost* (or a *payoff*).
- A *solution* is a sequence of moves from the initial state to a goal state with *minimum cost* (*maximum payoff*).
- Example: route finding (states are cities and moves are directly connecting roads), but it also applies to CAs ...

A *search algorithm* defines the order in which to traverse the tree:

- Uninformed search: breadth-first, depth-first, iterat. deepening
- Heuristic-guided search: branch-and-bound, A*

State Space and Moves

There are (at least) two natural ways of representing the state space and moves between states:

- Either: Define a state as a set of *goods* for which an allocation decision has already been made. Then making a move in the state space amounts to making a decision for a further good.
- Or: Define a state as a set of atomic *bids* for which an acceptance decision has already been made. In this case, a move amounts to making a decision for a further bid.

What is the initial state? What are the goal states?

According to Sandholm (2006), the bid-oriented approach tends to give better performance in practice.

Moves in Bid-oriented Search

We represent *bids* as triples (a_i, B_i, p_i): agent a_i is offering to buy the bundle B_i for a price of p_i.

The *initial state* is when no decisions on bids have been made.

A *move* amounts to making a decision (accept/reject) for a new bid.

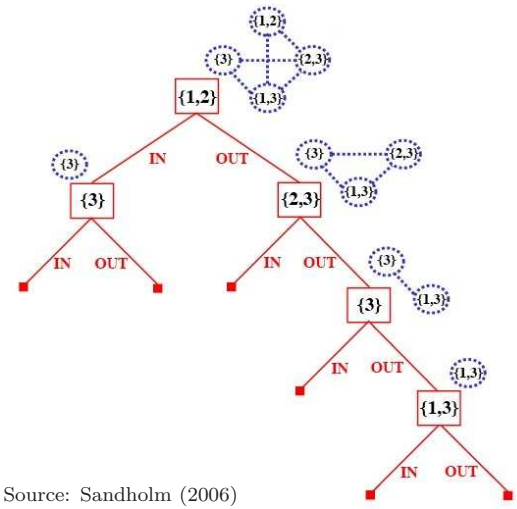
The *bidding language* specifies which bids (if any) *must* be accepted/rejected given earlier decisions. Examples:

- For both the OR and the XOR language: only accept bids with *empty intersection of bundles*.
- For the XOR language: accept *at most one bid per agent*.

We are in a *goal state* once a decision for every bid has been made (some of which will be consequences of the explicit choices).

Observe that that the search tree will be *binary* (accept or reject?).

Example



Source: Sandholm (2006)

Uninformed Search

Uninformed search algorithms (in particular depth-first search) can be used to find a solution with a given minimum revenue: traverse the tree and keep the best solution encountered so far in memory.

Optimality can only be guaranteed if we traverse the entire search tree (not feasible for interesting problem instances).

Heuristic-guided Search

In the worst case, any algorithm may have to search the entire search tree. But good *heuristics*, that tell us which part of the tree to explore next, often allow us to avoid this in practice.

For any node N in the search tree, let $g(N)$ be the revenue generated by accepting (only) the bids accepted according to N.

We are going to need a heuristic that allows us to estimate for every node N how much revenue over and above $g(N)$ can be expected if we pursue the path through N. This will be denoted as $h(N)$. The more accurate the estimate, the better — but the only strict requirement is that h *never underestimates* the true revenue.

We are going to describe two algorithms using such heuristics:

- *depth-first branch-and-bound*
- *the A* algorithm*

Heuristic Upper Bounds on Revenue

Sandholm (2006) discusses several ways of defining a heuristic function h such that $g(N) + h(N)$ is guaranteed to be an upper bound on revenue for any path through node N.

Here is one such heuristic function:

- For each good g, compute its *maximum contribution* as:

$$c(g) \quad = \quad \max\{\frac{p}{|B|} \mid (B,p) \in \mathit{Bids} \text{ and } g \in B\}$$

- Then define $h(N)$ as the sum of all $c(g)$ for those goods g that have not yet been allocated in N.

This is indeed an upper bound (why?).

Depth-first Branch-and-Bound

This algorithm works like basic (uninformed) depth-first search, except that branches that have no chance of developing into an optimal solution get pruned on the fly:

- Traverse the search tree in *depth-first* order.
- Keep track of which of the nodes encountered so far would generate maximum revenue. Call that node N^*.
- If a node N with $g(N) + h(N) \leq g(N^*)$ is encountered, *remove* that node and all its offspring from the search tree.

This is *correct* (guarantees that the optimal solution does not get removed) whenever the heuristic function h is guaranteed never to underestimate expected marginal revenue (why?).

The A* Algorithm

The *A* algorithm* (Hart *et al.*, 1968) is probably the most famous search algorithm in AI. It works as follows:

- The *fringe* is the set of leaf nodes of the subtree visited so far (initially just the root node).
- Compute $f(N) = g(N) + h(N)$ for every node N in the fringe.
- Expand the node N *maximising* $f(N)$; that is, remove it from the fringe and add its (two) immediate children instead.

By a standard result in AI, A* with an *admissible* heuristic function (here: h never underestimates marginal revenue) is *optimal:* the first solution found (when no bids are left) will generate maximum revenue.

P. Hart, N. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

Branching Heuristics

So far, we have not specified *which bid* to select for branching in each round (for any of our algorithms). This choice does not affect correctness, but it may affect speed.

There are two basic heuristics for bid selection:

- Select a bid with a *high price* and a *low number of items*.
- Select a bid that would *decompose* the conflict graph of the remaining bids (if available).

Tractable Subproblems

As a final example for possible fine-tuning of the algorithm, we can try to *identify tractable subproblems* at nodes and solve them using *special-purpose algorithms*.

Here are two very simple examples:

- If the bid conflict graph is *complete*, i.e., any pair of remaining bids is in conflict, then only one of them can be accepted.
  $\rightsquigarrow$ Simply pick the one with the *highest price*.
- If the bid conflict graph has *no edges*, then there is no conflict between any of the remaining bids.
  $\rightsquigarrow$ Accept *all* remaining bids (assuming positive prices).

Summary

- Quick introduction to *basic auction theory:* English, Dutch, first-price sealed-bid, and Vickrey auctions; revenue equivalence
- Combinatorial auctions are mechanisms to allocate a number of indivisible goods to a number of agents.
- Winner determination in CAs is *NP-complete* (in general).
- We have seen both *special cases* that are still NP-complete, and others that are tractable.
- The WDP can be solved by off-the-shelf *integer programming* tools as well as specialised *AI search techniques*.
- Our criterion for optimality has been *maximum revenue*. Alternatively, we could try to optimise wrt. a social welfare ordering (observe that revenue and utilitarian social welfare coincide in case bidders submit true and complete valuations).

References

The main reference on combinatorial auctions is the handbook edited by Cramton, Shoham, and Steinberg:

- P. Cramton, Y. Shoham, and R. Steinberg (eds.). *Combinatorial Auctions*. MIT Press, 2006.

Of particular relevance to this lecture are the following two chapters:

- D. Lehmann, R. Müller, and T. Sandholm. *The Winner Determination Problem* (Chapter 12).
- T. Sandholm. *Optimal Winner Determination Algorithms* (Chapter 14).

The paper by Rothkopf *et al.* on tractable instances of the WDP is one of the earliest examples of work on the computational aspects of CAs:

- M.H. Rothkopf, A. Pekeč, and R.M. Harstad. Computationally Manageable Combinational Auctions. *Management Science*, 44(8):1131–1147, 1998.

What next?

Today we have looked into the *computational* and the *algorithmic* aspects of combinatorial auctions. Next week we are going to deal with the *game-theoretical* side of combinatorial auctions:

- *Mechanism Design*