# Bidding Languages and Winner Determination for Mixed Multi-unit Combinatorial Auctions

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

[ joint work with Jesús Cerquides, Andrea Giovannucci, and Juan A. Rodríguez-Aguilar (all Barcelona) ]

## Talk Outline

- Introduction to Combinatorial Auctions
- The Mixed Auction Model
- Bidding Languages
- Winner Determination
- Conclusions

## Auctions

- Simple protocols to decide which agent should receive a particular good an how much they should pay for it
- English auction, Dutch auction, sealed-bid auction, ...
- Interesting game-theoretical questions: how do we stop agents from strategising and make sure they report their true preferences? ($\rightsquigarrow$ but not today)

# Combinatorial Auctions

In a *combinatorial auction*, the auctioneer puts several goods on sale and the other agents submit bids for entire bundles of goods.

Given a set of bids, the *winner determination problem* (WDP) is the problem of deciding which of the bids to accept.

- The solution must be *feasible* (no good may be allocated to more than one agent).
- Ideally, it should also be *optimal* (in the sense of maximising revenue for the auctioneer).

Clearly, finding an optimal solution to the WDP can be tricky (it's NP-complete, actually).

So besides the game-theoretical problem of stopping bidders from strategising, in combinatorial auctions we also face a challenging *algorithmic* problem.

# Communicating Bids

Suppose we are running a combinatorial auction with $n$ goods. So there are $2^n - 1$ bundles that agents may want to bid for.

For interesting values of $n$, it is not possible to communicate your valuations to the auctioneer by simply stating your price for each and every bundle. $\rightsquigarrow$ How do we best communicate/represent preferences in combinatorial domains?

For combinatorial auctions, this is the job of the *bidding language*.

Example: Bid for a small number of bundles with the implicit understanding that you will honour any combination of bids that is feasible and pay the sum of the associated prices (*OR-language*).

# Example

Each bidder submits a number of bids describing their valuation. Each bid $(B, p)$ specifies which price $p$ the bidder is prepared to pay for a particular bundle $B$. The auctioneer may accept at most one atomic bid per bidder (*XOR-language*).

Agent 1: $(\{a, b\}, 5)$ XOR $(\{b, c\}, 7)$ XOR $(\{c, d\}, 6)$

Agent 2: $(\{a, d\}, 7)$ XOR $(\{a, c, d\}, 8)$

Agent 3: $(\{b\}, 5)$ XOR $(\{a, b, c, d\}, 12)$

What would be the optimal solution?

The importance of CAs has been recognised for quite some time (in Economics), but only very recently have algorithms that can solve realistic problem instances been developed (in Computer Science).

## Types of Combinatorial Auctions

- *Direct auction:* the auctioneer is selling goods to the bidders
- *Reverse auction:* the auctioneer is buying goods from the bidders
- *Multi-unit auction:* there may be several copies of each (type of) good available ($\leadsto$ bundles are multisets)

# Talk Outline (again)

- Introduction to Combinatorial Auctions ✓
- The Mixed Auction Model
- Bidding Languages
  - Semantics of the Basic Language
  - Additional Features for Succinctness
  - Expressive Power
- Winner Determination
  - Problem Definition
  - Computational Complexity
  - Integer Programming Solution
- Conclusions

# Mixed Auctions

- Suppose the auctioneer would like to both sell and buy goods: combine *direct* and *reverse* auctions.
- The auctioneer may be able to *transform* goods: instead of buying a car he may choose to buy certain components and build the car by himself (at a cost).
- Generalising further, the auctioneer may even solicit *bids for transformations*.
- We call the resulting model *mixed auctions*. In principle, both single-unit and multi-unit variants of this model make sense, but I shall only speak about the more interesting mixed *multi-unit* combinatorial auctions . . .

# Transformations

Let $G$ be a finite set of types of goods.

A *transformation* is a pair of multisets over $G$: $(\mathcal{I}, \mathcal{O}) \in \mathbb{N}^G \times \mathbb{N}^G$

"I can deliver $\mathcal{O}$ *after* having received $\mathcal{I}$."

Interesting special cases are where either $\mathcal{I} = \{\}$ or $\mathcal{O} = \{\}$.

Bidders will be able to offer several such transformations; that is agents will negotiate over *multisets of transformations* $\mathcal{D} \in \mathbb{N}^{(\mathbb{N}^G \times \mathbb{N}^G)}$.

Example: $\{(\{\}, \{a\}), (\{b\}, \{c\})\}$ means that the agent in question is able to deliver $a$ (no input required) and that it is able to deliver $c$ if provided with $b$. Note that this is not the same as $\{(\{b\}, \{a, c\})\}$. In the former case, if another agent is able to produce $b$ if provided with $a$, we can get $c$ from nothing; in the latter case this would not work.

# Subsumption

Define $(\mathcal{I}, \mathcal{O}) \sqsubseteq (\mathcal{I}', \mathcal{O}')$ iff $\mathcal{I} \subseteq \mathcal{I}'$ and $\mathcal{O} \supseteq \mathcal{O}'$ ("no worse than").

This can be extended to multisets of transformations as follows:

**Definition 1 (Subsumption)** *Let $\mathcal{D}, \mathcal{D}' \in \mathbb{N}^{(\mathbb{N}^G \times \mathbb{N}^G)}$. We say that $\mathcal{D}$ is subsumed by $\mathcal{D}'$ ($\mathcal{D} \sqsubseteq \mathcal{D}'$) iff:*

$(i)$ *$\mathcal{D}$ and $\mathcal{D}'$ have the same cardinality: $|\mathcal{D}| = |\mathcal{D}'|$.*

$(ii)$ *There exists a surjective mapping $f : \mathcal{D} \to \mathcal{D}'$ such that, for all transformations $t \in \mathcal{D}$, we have $t \sqsubseteq f(t)$.*

Example: Using a simplified notation for the innermost sets, we have $\{(a, bb), (cc, dd)\} \sqsubseteq \{(cc, d), (aaa, b)\}$.

# Valuations

We use *valuations* $v : \mathbb{N}^{(\mathbb{N}^G \times \mathbb{N}^G)} \to \mathbb{R}$ to describe how much an agent values any given multiset of transformations.

- $v(\mathcal{D}) = p$ means that the agent with valuation $v$ is willing to *pay* $p$ in return for being allocated all the transformations in $\mathcal{D}$
- For $p < 0$ the agent expects to *receive* a payment of $|p|$.
  Example: $v(\{(\{oven, dough\}, \{oven, cake\})\}) = -20$
- Write $v(\mathcal{D}) = \perp$ to say that $v$ is *undefined* over $\mathcal{D}$ (the agent *cannot* accept the transformations).

Valuations will often be normalised and monotonic:

**Definition 2 (Normalised valuation)** *A valuation $v$ is normalised iff $v(\mathcal{D}) = 0$ whenever $\mathcal{I} = \mathcal{O}$ for all $(\mathcal{I}, \mathcal{O}) \in \mathcal{D}$.*

**Definition 3 (Monotonic valuation)** *A valuation $v$ is monotonic iff $v(\mathcal{D}) \leq v(\mathcal{D}')$ whenever $\mathcal{D} \sqsubseteq \mathcal{D}'$.*

## Atomic Bids

An *atomic bid* $\textsc{bid}(\{(\mathcal{I}^1, \mathcal{O}^1), \ldots, (\mathcal{I}^n, \mathcal{O}^n)\}, p)$ specifies a finite multiset of finite transformations and a price.

Under the assumption of *free disposal* at the bidder's side, the bid $Bid = \textsc{bid}(\mathcal{D}, p)$ defines the following valuation:

$$v_{Bid}(\mathcal{D}') = \begin{cases} p & \text{if } \mathcal{D} \sqsubseteq \mathcal{D}' \\ \bot & \text{otherwise} \end{cases}$$

To obtain the valuation function defined by $Bid$ without the free disposal assumption, simply replace $\sqsubseteq$ by equality.

# The XOR-Language

A suitable *bidding language* should allow a bidder to encode choices between alternative bids and the like.

In the *XOR-language* we only allow XOR-combinations of atomic bids. The intuitive semantics is that at most one of the atomic bids of each bidder can be selected by the auctioneer. Formal semantics:

$$v_{Bid}(\mathcal{D}) \;=\; \max\{v_{Bid_i}(\mathcal{D}) \mid i \in \{1..n\}\}$$

where $Bid = Bid_1 \text{ XOR } \cdots \text{ XOR } Bid_n$

That is, XOR simply selects the atomic bid corresponding to the valuation giving maximum profit for the auctioneer.

# Full Bidding Language

We can define several other interesting language constructs. While these can greatly improve *succinctness* of representation, they do not increase *expressive power*.

To make this point, we present them as rewrite rules, allowing us to translate from the full bidding language into the XOR-language:

$$
\begin{aligned}
X \text{ OR } Y &\leadsto X \text{ XOR } Y \text{ XOR } (X \text{ AND } Y) \\
X \text{ IMPLIES } Y &\leadsto (X \text{ AND } Y) \text{ XOR } Y \\
(X \text{ XOR } Y) \text{ AND } Z &\leadsto (X \text{ AND } Z) \text{ XOR } (Y \text{ AND } Z) \\
\text{BID}(\mathcal{D}, p) \text{ AND BID}(\mathcal{D}', p') &\leadsto \text{BID}(\mathcal{D} \cup \mathcal{D}', p + p')
\end{aligned}
$$

Standard definitions of OR require checking feasibility. This makes little sense for multi-unit auctions, so checking feasibility is not part of the semantics of the bidding language (but is left to the auctioneer).

## Example

Write this to say that up to $n$ copies of the same $Bid$ are acceptable:

$$Bid^{\leq n} \quad = \quad \underbrace{(Bid \text{ OR } \cdots \text{ OR } Bid)}_{n \text{ times}}$$

To say that exactly $n$ copies of $Bid$ need to be selected together, write:

$$Bid^{n} \quad = \quad \underbrace{(Bid \text{ AND } \cdots \text{ AND } Bid)}_{n \text{ times}}$$

Now I can express that I'm prepared to buy up to 50 items of type $a$ for €$25$ each, and then up to 100 more for €$20$ each:

$$[(a, 20)^{\leq 100} \text{ IMPLIES } (a, 25)^{50}] \text{ XOR } (a, 25)^{\leq 50}$$

# Expressive Power

**Definition 4 (Finitely-peaked val.)** *A valuation $v$ is finitely-peaked iff $v$ is only defined over finite multisets of pairs of finite multisets and $\{\mathcal{D} \in \mathbb{N}^{(\mathbb{N}^G \times \mathbb{N}^G)} \mid v(\mathcal{D}) \neq \bot\}$ is finite.*

**Proposition 1 (No free disposal)** *The XOR-language without free disposal can represent all finitely-peaked valuations, and only those.*

**Definition 5 (Monotonic closure)** *The monotonic closure $\hat{v}$ of a valuation $v$ is defined as $\hat{v}(\mathcal{D}) = \max\{v(\mathcal{D}') \mid \mathcal{D}' \sqsubseteq \mathcal{D}\}$.*

**Proposition 2 (Free disposal)** *The XOR-language with free disposal can represent all valuations that are the monotonic closure of a finitely-peaked valuation, and only those.*

# WDP: Informal Definition

The *input* to the winner determination problem (WDP) consists of the *bids* submitted by the bidders, a multiset $\mathcal{U}_{in}$ of goods the auctioneer holds initially, and a multiset $\mathcal{U}_{out}$ he is expected to end up with.

A *solution* will be a *sequence* (not a set!) of selected transformations. A *valid* solution has to meet two kinds of conditions:

(1) *Bidder constraints:* The transformations selected have to respect the bids submitted. Say, if a bidder submits an XOR-combination of transformations, at most one of them may be accepted.

(2) *Auctioneer constraints:* The sequence has to be implementable: (a) check that $\mathcal{U}_{in}$ is a superset of the input of the first transformation; (b) then update the set of goods held by the auctioneer after each transformation and check that it is a superset of the input of the next one; (c) finally check that the auctioneer holds at least $\mathcal{U}_{out}$ in the end.

An *optimal* solution is a valid solution that maximises the sum of prices associated with the atomic bids selected.

## Computational Complexity

Winner determination is NP-complete for standard CAs (shown via a reduction from SET PACKING). Although intuitively more complicated than standard CAs, mixed auctions are *no worse* than that:

**Proposition 3 (Complexity)** *Checking whether there exists a valid solution with revenue* $\geq K$ *is NP-complete for mixed auctions.*

*Proof.* Mixed auctions can simulate standard CAs $\Rightarrow$ *NP-hardness*. ✓ *NP-membership* follows from the fact that we can verify a supposed solution in polynomial time. All that is required is checking that the proposed solution is valid and then add up the prices. ✓ □

# Integer Programming

- NP-completeness means that there will be no general method to solve the WDP efficiently.
- However, in such situations, one can often make use of highly optimised mathematical programming software and may get good performance even for large problem instances.
- To do this, we have to encode the WDP as an integer program:
  - Introduce a number of integer (or binary) decision variables.
  - State the optimisation problem as the problem of maximising the value of some linear expression involving these variables.
  - Formulate any number of side constraints as (in)equations over linear expressions involving these same variables.

# Standard Combinatorial Auctions

For standard CAs it's straightforward to formulate the WDP as an integer program. As an example, we give the WDP for direct single-unit combinatorial auctions using the OR-language.

Let $(B_{ij}, p_{ij})$ be the *$j$th atomic bid* within the OR-bid submitted by the *$i$th bidder*.

Introduce binary decision variables $x_{ij}$, where $x_{ij} = 1$ iff $(B_{ij}, p_{ij})$ is selected by the auctioneer.

Now the WDP can be stated as follows:

$$\max \sum_{ij} x_{ij} \cdot p_{ij} \text{ subject to } \sum_{ij} x_{ij} \cdot |\{g\} \cap B_{ij}| \leq 1 \text{ for all goods } g$$

That is, the auctioneer will try to maximise the sum of the prices of the accepted atomic bids, whilst ensuring that no good gets allocated to more than one agent.

# WDP: Integer Programming Formulation

For mixed auctions, the solution will be a *sequence* of transformations.

Introduce binary decision variables $x^m_{ijk}$, where $x^m_{ijk} = 1$ iff the *$k$th transformation* in the *$j$th atomic bid* of the *$i$th bidder* is selected for the *$m$th position* in the sequence.

Note that the overall number of transformations in the bids is an upper bound for $m$ (so we know how many of these variables to create).

Further binary decision variables used:

- $x^m = 1$ iff any transformation at all is selected for position $m$
- $x_{ijk} = 1$ iff transformation $t_{ijk}$ is selected at all
- $x_{ij} = 1$ iff any transformation from the $j$th atomic bid of the $i$th bidder is selected

Next we define several constraints over these variables and then state the actual optimisation problem ...

# Bidder Constraints

For simplicity, we restrict ourselves to the XOR-language (which is as expressive as the full language). Encoding the constraints imposed by other operations is not difficult and can greatly improve efficiency.

(1) Respect the BID-operator. Either select all transformations belonging to a particular atomic bid, or none of them:

$$\forall ijk: \quad x_{ij} = x_{ijk}$$

(2) Respect the XOR-operator. Accept at most one atomic bid from each bidder:

$$\forall i: \quad \sum_j x_{ij} \leq 1$$

## Sequence Well-formedness Constraints

(3) Each transformation can be selected at most once for inclusion anywhere in the sequence:

$$\forall ijk: \quad x_{ijk} = \sum_{m} x_{ijk}^{m}$$

(4) At most one transformation can be selected for any one position in the sequence:

$$\forall m: \quad x^{m} = \sum_{ijk} x_{ijk}^{m}$$

(5) There should be no gaps in the sequence:

$$\forall m: \quad x^{m} \geq x^{m+1}$$

# Implementability Constraint

(6) The multiset of goods held by the auctioneer just before the $m$th transformation must be a superset of the input required by that transformation.

To this end, we introduce integer decision variables $\mathcal{M}^m(g)$ for each $m$ and each good $g$ to represent how many copies of $g$ the auctioneer holds after the $m$th transformation:

$$\forall m g: \quad \mathcal{M}^m(g) = \mathcal{U}_{in}(g) + \sum_{\ell=1}^{m} \sum_{ijk} x_{ijk}^{\ell} \cdot (\mathcal{O}_{ijk}(g) - \mathcal{I}_{ijk}(g))$$

Here $\mathcal{U}_{in}(g)$ is the number of copies of good $g$ held by the auctioneer initially.

Now we can write down the implementability constraint:

$$\forall m g: \quad \mathcal{M}^{m-1}(g) \geq \sum_{ijk} x_{ijk}^{m} \cdot \mathcal{I}_{ijk}(g)$$

## Output Constraint

(7) After having implemented all the selected transformation, the goods held by the auctioneer should form a superset of the required output:

$$\forall g: \quad \mathcal{M}^{|T|}(g) \geq \mathcal{U}_{out}(g)$$

Here $\mathcal{U}_{out}(g)$ is the number of copies of good $g$ the auctioneer is expected to end up with; and $|T|$ is the overall number of transformations mentioned anywhere in the bids.

Observe that $\mathcal{M}^{|T|}$ (the set held at the end of the fake sequence including many empty positions at the end) is the same as the multiset held right after the last real transformation.

Note that the above is the constraint for the variant of the WDP with *free disposal* at the side of the auctioneer.

## Optimisation

Now solving the WDP amounts to solving this integer program:

$$\max \sum_{ij} x_{ij} \cdot p_{ij} \quad \text{subject to constraints (1)–(7)}$$

Here $p_{ij}$ is the price associated with the $j$th atomic bid of the $i$th bidder. That is, we are maximising the sum of the prices associated with the atomic bids selected.

## Discussion

- Main challenge: We had to encode the fact that the solution has to form a *sequence*, which is not the case for standard CAs ...
- Critical issue: The number of decision variables required is *quadratic* in the number of atomic bids (linear for standard CAs). To be seen what this does to performance ...
- Is it possible to get around the quadratic number of variables? A vague idea would be to try using *integer decision variables* $x_{ijk} = m$ to encode that transition $t_{ijk}$ takes position $m$ ...

# Conclusions

- A new auction model: (1) bidding for *transformations* of goods; (2) a solution is a *sequence* rather than a set of atomic bids
- Generalises all of single/multi-unit direct/reverse combinatorial auctions, combinatorial exchanges, ...
- *Bidding Languages:* expressive completeness results with respect to finitely-peaked valuations
- *Winner Determination:* no increase in (theoretical) complexity; Integer Programming formulation difficult but possible
- *Mechanism Design:* the standard result on strategy-proofness of the VCG mechanism applies also to mixed auctions

Jesús Cerquides, Ulle Endriss, Andrea Giovannucci, and Juan A. Rodríguez-Aguilar. *Bidding Languages and Winner Determination for Mixed Multi-unit Combinatorial Auctions*. Proc. 20th Intl. Joint Conference on Artificial Intelligence (IJCAI-2007).