Bidding Languages

Noam Nissan

October 18, 2004

Presenter:

Shahram Esmaeilsabzali

Outline

Outline

- The Problem

Outline

- The Problem
- Some Bidding Languages(OR, XOR, and etc)

Outline

- The Problem
- Some Bidding Languages(`OR`, `XOR`, and etc)
- Extension to Bidding Languages

Outline

- The Problem
- Some Bidding Languages(`OR`, `XOR`, and etc)
- Extension to Bidding Languages
- Complexity of handling bids

Outline

- The Problem
- Some Bidding Languages(`OR`, `XOR`, and etc)
- Extension to Bidding Languages
- Complexity of handling bids
- Conclusion

What is a bid?

What is a bid?

"Theoretically speaking, bids are simply abstract elements drawn from some space of strategies defined by the auction." [Nis04]

What is a bid?

"Theoretically speaking, bids are simply abstract elements drawn from some space of strategies defined by the auction." [Nis04]

- Main challenge arises when dealing with combinational auctions.

What is a bid?

"Theoretically speaking, bids are simply abstract elements drawn from some space of strategies defined by the auction." [Nis04]

- Main challenge arises when dealing with combinational auctions.
- For m items, there are $2^m - 1$ possible bids to specify.

What is a bid?

"Theoretically speaking, bids are simply abstract elements drawn from some space of strategies defined by the auction." [Nis04]

- Main challenge arises when dealing with combinational auctions.
- For m items, there are $2^m - 1$ possible bids to specify.
- Bidding languages are meant to provide the *syntax* for encoding bids' information in a succinct, simple manner.

What is a bid?

"Theoretically speaking, bids are simply abstract elements drawn from some space of strategies defined by the auction." [Nis04]

- Main challenge arises when dealing with combinational auctions.
- For m items, there are $2^m - 1$ possible bids to specify.
- Bidding languages are meant to provide the *syntax* for encoding bids' information in a succinct, simple manner.
- Needless to say, similar to any language, there is a trade of between expressiveness and simplicity.

Some Definitions

Some Definitions

- We denote the number of items in an auction by m.

Some Definitions

- We denote the number of items in an auction by m.
- v is the valuation function. $v(S)$ is the valuation of items in S by *a* bidder.

Some Definitions

- We denote the number of items in an auction by m.
- v is the valuation function. $v(S)$ is the valuation of items in S by *a* bidder.
- Notice that the value for S is assumed to be equal to $v(S)$. In this sense, the bidding languages can be considered *valuation languages.*

Some Definitions

- We denote the number of items in an auction by m.

- v is the valuation function. $v(S)$ is the valuation of items in S by *a* bidder.

- Notice that the value for S is assumed to be equal to $v(S)$. In this sense, the bidding languages can be considered *valuation languages.*

- **Complementary:** if $v(S \cup T) > v(S) + v(T)$.

Some Definitions

- We denote the number of items in an auction by m.
- v is the valuation function. $v(S)$ is the valuation of items in S by *a* bidder.
- Notice that the value for S is assumed to be equal to $v(S)$. In this sense, the bidding languages can be considered *valuation languages.*

- **Complementary:** if $v(S \cup T) > v(S) + v(T)$.
- **Substitutes:** if $v(S \cup T) < v(S) + v(T)$.

Valuation Functions

Valuation Functions

Bidding languages basically try to efficiently model different *patterns* for bids. We are interested in bidding languages that can present useful patterns efficiently.

Valuation Functions

Bidding languages basically try to efficiently model different *patterns* for bids. We are interested in bidding languages that can present useful patterns efficiently.

A natural taxonomy for valuation functions is:

Valuation Functions

Bidding languages basically try to efficiently model different *patterns* for bids. We are interested in bidding languages that can present useful patterns efficiently.

A natural taxonomy for valuation functions is:

1. Symmetric Valuations: The valuation function $v(S)$ depends only on $|S|$. All items are similar from bidders' point of view.

Valuation Functions

Bidding languages basically try to efficiently model different *patterns* for bids. We are interested in bidding languages that can present useful patterns efficiently.

A natural taxonomy for valuation functions is:

1. Symmetric Valuations: The valuation function $v(S)$ depends only on $|S|$. All items are similar from bidders' point of view.

2. Asymmetric Valuations: The valuation function distinguishes between different items in the auction.

Some Symmetric Valuations

Some Symmetric Valuations

- **The simple additive valuation:** $v(S) = |S|$

Some Symmetric Valuations

- **The simple additive valuation:** $v(S) = |S|$

- **The simple unit demand valuation:** $v(S) = 1$ for all $S \neq \emptyset$.

Some Symmetric Valuations

- **The simple additive valuation:** $v(S) = |S|$
- **The simple unit demand valuation:** $v(S) = 1$ for all $S \neq \emptyset$.
- **The simple K-Budget valuation:** $v(S) = \texttt{min}(K, |S|)$.

Some Symmetric Valuations

- **The simple additive valuation:** $v(S) = |S|$
- **The simple unit demand valuation:** $v(S) = 1$ for all $S \neq \emptyset$.
- **The simple K-Budget valuation:** $v(S) = \mathtt{min}(K, |S|)$.
- **The Majority-valuation:** $v(S) = 1$ if $|S| \geq m/2$ else $v(S) = 0$.

Some Symmetric Valuations

- **The simple additive valuation:** $v(S) = |S|$
- **The simple unit demand valuation:** $v(S) = 1$ for all $S \neq \emptyset$.
- **The simple K-Budget valuation:** $v(S) = \texttt{min}(K, |S|)$.
- **The Majority-valuation:** $v(S) = 1$ if $|S| \geq m/2$ else $v(S) = 0$.
- **The General Symmetric valuation:** The price p_j specifies the price for the j'th item won and $v(S) = \sum_{j=1}^{|S|} p_j$.

Some Symmetric Valuations

- **The simple additive valuation:** $v(S) = |S|$
- **The simple unit demand valuation:** $v(S) = 1$ for all $S \neq \emptyset$.
- **The simple K-Budget valuation:** $v(S) = \mathtt{min}(K, |S|)$.
- **The Majority-valuation:** $v(S) = 1$ if $|S| \geq m/2$ else $v(S) = 0$.
- **The General Symmetric valuation:** The price p_j specifies the price for the j'th item won and $v(S) = \sum_{j=1}^{|S|} p_j$.
- **A Downward Symmetric valuation:** A symmetric valuation where $p_1 \geq p_2 \geq \ldots \geq p_m$.

Some Asymmetric Valuations

Some Asymmetric Valuations

- **An additive valuation:** Value of item j is v^j and $v(S) = \Sigma_{j \in S} v^j$.

Some Asymmetric Valuations

- **An additive valuation:** Value of item j is v^j and $v(S) = \Sigma_{j \in S} v^j$.
- **The unit demand valuation:** The value for item j is v^j and the bidder wants only a single item, $v(S) = \mathtt{max}_{j \in S} v^j$.

Some Asymmetric Valuations

- **An additive valuation:** Value of item j is v^j and $v(S) = \Sigma_{j \in S} v^j$.
- **The unit demand valuation:** The value for item j is v^j and the bidder wants only a single item, $v(S) = \mathtt{max}_{j \in S} v^j$.
- **The monochromatic valuation:** There are colors `blue` and `red`, and the bidder requires items from the "same" color. Given the value for each item is 1, the valuation for any set of k `blue` and l `red` items is $\mathtt{max}(k, l)$.

Some Asymmetric Valuations

- **An additive valuation:** Value of item j is v^j and $v(S) = \Sigma_{j \in S} v^j$.
- **The unit demand valuation:** The value for item j is v^j and the bidder wants only a single item, $v(S) = \mathtt{max}_{j \in S} v^j$.
- **The monochromatic valuation:** There are colors `blue` and `red`, and the bidder requires items from the "same" color. Given the value for each item is 1, the valuation for any set of k `blue` and l `red` items is $\mathtt{max}(k, l)$.
- **The one-of-each-kind valuation:** There are k pairs and l singletons $(|S| = 2k + l)$. the bidder desires one item from each type at value 1. The valuation would be then $(k + l)$.

Bidding Languages

Bidding Languages

- Having recognized some natural valuation patterns, we would like to have *bidding languages* that can specify such patterns efficiently and succinctly.

Bidding Languages

- Having recognized some natural valuation patterns, we would like to have *bidding languages* that can specify such patterns efficiently and succinctly.

- Often each language is good in expressing some patterns and weak or unable in expressing some other patterns.

Bidding Languages

- Having recognized some natural valuation patterns, we would like to have *bidding languages* that can specify such patterns efficiently and succinctly.

- Often each language is good in expressing some patterns and weak or unable in expressing some other patterns.

- While some classes of languages can be compared based on their expressivity, it is not always possible to accurately compare two bidding languages.

Atomics Bids

Atomics Bids

- Also called *Single Minded Bids*. [LOS99]

Atomics Bids

- Also called *Single Minded Bids*. [LOS99]

- (S, P) says that the bidder chooses S, a subset of available items, for price P:

$$v(S) = P$$

Atomics Bids

- Also called *Single Minded Bids*. [LOS99]

- (S, P) says that the bidder chooses S, a subset of available items, for price P:
$$v(S) = P$$

- For all $S' \neq S$, $v(S') = 0$.

Atomics Bids

- Also called *Single Minded Bids*. [LOS99]

- (S, P) says that the bidder chooses S, a subset of available items, for price P:
$$v(S) = P$$

- For all $S' \neq S$, $v(S') = 0$.

- Obviously, atomic bids are not expressive enough to express majority of symmetric/assymetric valuation patterns, e.g. they can not specify "simple additive valuation".

OR Bids

OR Bids

- Bidder chooses multiple bids. Bids are not necessarily disjoint.

$$S = (S_1, p_1)\, \texttt{OR}\, (S_2, p_2)\, \texttt{OR}\, \ldots\, \texttt{OR}\, (S_k, p_k)$$

OR Bids

- Bidder chooses multiple bids. Bids are not necessarily disjoint.

$$S = (S_1, p_1)\,\texttt{OR}\,(S_2, p_2)\,\texttt{OR}\,\ldots\,\texttt{OR}\,(S_k, p_k)$$

- The valuation of bids for a certain bidder consists of only disjoint bids.

$$v(S) = \texttt{Max}_{\texttt{W}}(\textstyle\sum_{i \in W} p_i), \quad S_i, S_j \in W \Rightarrow S_i \cap S_j = \emptyset$$

OR Bids

- Bidder chooses multiple bids. Bids are not necessarily disjoint.

$$S = (S_1, p_1)\,\texttt{OR}\,(S_2, p_2)\,\texttt{OR}\,\ldots\,\texttt{OR}\,(S_k, p_k)$$

- The valuation of bids for a certain bidder consists of only disjoint bids.

$$v(S) = \texttt{Max}_{\texttt{W}}(\textstyle\sum_{i\in W} p_i), \quad S_i, S_j \in W \Rightarrow S_i \cap S_j = \emptyset$$

- The real problem with OR bids is that they don't support substitutability.

OR Bids: Example

OR Bids: Example

- It is not to possible to express *simple unit demand valuation*.

OR Bids: Example

- It is not to possible to express *simple unit demand valuation*.
- Consider the following `OR` bid:

$$S = \{\, (\{1\}, 5\$) \,\texttt{OR}\, (\{2\}, 4\$) \,\texttt{OR}\, (\{1,2\}, 7\$)) \,\}$$

OR Bids: Example

- It is not to possible to express *simple unit demand valuation*.
- Consider the following `OR` bid:

$$S = \{\, (\{1\}, 5\$) \,\texttt{OR}\, (\{2\}, 4\$) \,\texttt{OR}\, (\{1, 2\}, 7\$)) \,\}$$

The bidder is interested in having both items 1 and 2, only if she pays 7$.

OR Bids: Example

- It is not to possible to express *simple unit demand valuation*.
- Consider the following OR bid:

$$S = \{\, (\{1\}, 5\$)\, \texttt{OR}\, (\{2\}, 4\$)\, \texttt{OR}\, (\{1,2\}, 7\$))\, \}$$

The bidder is interested in having both items 1 and 2, only if she pays 7$.

We can't express such constraints with OR bids.

XOR Bids

XOR Bids

XOR is expressive enough to express all valuation variations. Given a XOR bid:

$$S = (S_1, p_1)\,\texttt{XOR}\,(S_2, p_2)\,\texttt{XOR}\;\ldots\,\texttt{XOR}\,(S_k, p_k)$$

XOR Bids

XOR is expressive enough to express all valuation variations. Given a XOR bid:

$$S = (S_1, p_1)\,\texttt{XOR}\,(S_2, p_2)\,\texttt{XOR}\,\ldots\,\texttt{XOR}\,(S_k, p_k)$$

the valuation function is as follows:

$$v(S) = \texttt{max}_{i|s_i \subseteq S}\, p_i$$

XOR Bids: Cont'd

XOR Bids: Cont'd

- While `XOR` is more expressive than `OR`, there are valuations that can be specified more succinctly by `OR`.

XOR Bids: Cont'd

- While `XOR` is more expressive than `OR`, there are valuations that can be specified more succinctly by `OR`.

- Consider the simple additive valuation. `OR` can specify that in size m while `XOR` require 2^m clauses.

XOR Bids: Cont'd

- While XOR is more expressive than OR, there are valuations that can be specified more succinctly by OR.
- Consider the simple additive valuation. OR can specify that in size m while XOR require 2^m clauses.
- This motivates to combine OR and XOR. The result introduces two languages: OR-of-XORs and XOR-of-ORs.

OR-of-XORs

OR-of-XORs

- Bids consist of clauses that entirely consist of `XOR` bids and such clauses are connected by `OR`s.

$$(\ldots\,\texttt{XOR}\,\ldots)\;\texttt{OR}\;(\ldots\,\texttt{XOR}\,\ldots)\;\texttt{OR}\;\ldots\;\texttt{OR}\;(\ldots\,\texttt{XOR}\,\ldots)$$

OR-of-XORs

- Bids consist of clauses that entirely consist of `XOR` bids and such clauses are connected by `OR`s.

$$(\ldots \texttt{XOR} \ldots)\ \texttt{OR}\ (\ldots \texttt{XOR} \ldots)\ \texttt{OR}\ \ldots\ \texttt{OR}\ (\ldots \texttt{XOR} \ldots)$$

- A downward slopping symmetric evaluation can be expressed in size m^2.

$$p_1 \geq p_2$$
$$v(s) = (\ ((s, p_1)\ \texttt{XOR}\ (s, p_1))\ \texttt{OR}\ ((s, p_2)\ \texttt{XOR}\ (s, p_2))\ )$$

OR-of-XORs Cont'd

OR-of-XORs Cont'd

- The monochromatic valuation of m items, on the other hand, is exponential, $2.2^{m/2}$.

$$S = \{(s_{1r}, p_{1r}), (s_{2r}, p_{2r}), (s_{1b}, p_{1b}), (s_{2b}, p_{2b})\}$$

$$\begin{aligned} v(S) = ((s_{1r}, p_{1r})\,\mathtt{XOR}\,(s_{2r}, p_{2r})\,\mathtt{XOR}\,((s_{1r} \cup s_{2r}), p_{1r} + p_{2r}) \\ \mathtt{XOR}\ \ldots\ \mathtt{XOR}\,((s_{1b} \cup s_{2b}), p_{1b} + p_{2b}) \end{aligned}$$

OR-of-XORs Cont'd

- The monochromatic valuation of m items, on the other hand, is exponential, $2.2^{m/2}$.

$$S = \{(s_{1r}, p_{1r}), (s_{2r}, p_{2r}), (s_{1b}, p_{1b}), (s_{2b}, p_{2b})\}$$

$$\begin{aligned} v(S) = ((s_{1r}, p_{1r})\,\texttt{XOR}\,(s_{2r}, p_{2r})\,\texttt{XOR}\,((s_{1r} \cup s_{2r}), p_{1r} + p_{2r}) \\ \texttt{XOR} \ldots \texttt{XOR}\,((s_{1b} \cup s_{2b}), p_{1b} + p_{2b}) \end{aligned}$$

- This motivates the bidding language `XOR-of-ORs`.

XOR-of-ORs

XOR-of-ORs

- Bids consist of clauses that entirely consist of `OR` bids and such clauses are connected by `XOR`s.

$$(\ldots \texttt{OR} \ldots)\, \texttt{XOR}\, (\ldots \texttt{OR} \ldots)\, \texttt{XOR}\, \ldots\, \texttt{XOR}\, (\ldots \texttt{OR} \ldots)$$

XOR-of-ORs

- Bids consist of clauses that entirely consist of `OR` bids and such clauses are connected by `XOR`s.

$$(\ldots\,\texttt{OR}\ldots)\,\texttt{XOR}\,(\ldots\,\texttt{OR}\ldots)\,\texttt{XOR}\,\ldots\,\texttt{XOR}\,(\ldots\,\texttt{OR}\ldots)$$

- Consider the problem of specifying monochromatic valuation of m items. This can be achieved in size m with `XOR-of-ORs`.

$$S = \{(s_{1r}, p_{1r}), (s_{2r}, p_{2r}), (s_{1b}, p_{1b}), (s_{2b}, p_{2b})\}$$
$$v(S) = (\,((s_{1r}, p_{1r})\,\texttt{OR}\,(s_{2r}, p_{2r}))\,\texttt{XOR}\,((s_{1b}, p_{1b})\texttt{OR}(s_{2b}, p_{2b}))\,)$$

XOR-of-ORs

- Bids consist of clauses that entirely consist of `OR` bids and such clauses are connected by `XOR`s.

$$(\ldots \texttt{OR} \ldots)\,\texttt{XOR}\,(\ldots \texttt{OR} \ldots)\,\texttt{XOR}\,\ldots\,\texttt{XOR}\,(\ldots \texttt{OR} \ldots)$$

- Consider the problem of specifying monochromatic valuation of m items. This can be achieved in size m with `XOR-of-ORs`.

$$S = \{(s_{1r}, p_{1r}), (s_{2r}, p_{2r}), (s_{1b}, p_{1b}), (s_{2b}, p_{2b})\}$$
$$v(S) = (\,((s_{1r}, p_{1r})\,\texttt{OR}\,(s_{2r}, p_{2r}))\,\texttt{XOR}\,((s_{1b}, p_{1b})\texttt{OR}(s_{2b}, p_{2b}))\,)$$

- However, The K-budget valuation requires exponential time to be expressed.

OR/XOR Formulas

OR/XOR Formulas

- Alternatively, it is possible to specify the bids by applying `OR` and `XOR` on the valuation function.

OR/XOR Formulas

- Alternatively, it is possible to specify the bids by applying `OR` and `XOR` on the valuation function.

- This approach is also capable of simulating `XOR-of-ORs` and `OR-of-XORs`, and all other bidding languages discussed so far.

OR/XOR Formulas

- Alternatively, it is possible to specify the bids by applying `OR` and `XOR` on the valuation function.

- This approach is also capable of simulating `XOR-of-ORs` and `OR-of-XORs`, and all other bidding languages discussed so far.

  - $(v\,\mathtt{XOR}\,u)(S) = \mathtt{max}(v(S), u(S))$
  - $(v\,\mathtt{OR}\,u)(S) = \mathtt{max}_{R,T\subseteq S, R\cap T=\emptyset}(v(R) + u(T))$

OR/XOR Formulas

- Alternatively, it is possible to specify the bids by applying `OR` and `XOR` on the valuation function.

- This approach is also capable of simulating `XOR-of-ORs` and `OR-of-XORs`, and all other bidding languages discussed so far.

  – $(v \, \texttt{XOR} \, u)(S) = \texttt{max}(v(S), u(S))$
  – $(v \, \texttt{OR} \, u)(S) = \texttt{max}_{R,T \subseteq S, R \cap T = \emptyset}(v(R) + u(T))$

- `OR/XOR` formulas are defined recursively and provide succinct presentation of bids.

OR* Bids

$\mathbf{OR}^*$ Bids

- The idea is to express `XOR` with `OR`.

OR* Bids

- The idea is to express `XOR` with `OR`.
- This can be achieved by using dummy items.

$\mathbf{OR}^*$ Bids

- The idea is to express `XOR` with `OR`.
- This can be achieved by using dummy items.
- Dummy items don't have any values but can constrain bids to represent `XOR`.

$$(S_1, p_1)\, \texttt{XOR}\, (S_2, p_2)$$

$\mathbf{OR}^*$ Bids

- The idea is to express `XOR` with `OR`.
- This can be achieved by using dummy items.
- Dummy items don't have any values but can constrain bids to represent `XOR`.

$$(S_1, p_1)\,\texttt{XOR}\,(S_2, p_2)$$

Can be shown as:

$$(S_1 \cup \{dummy\}, p_1)\,\texttt{OR}\,(S_2 \cup \{dummy\}, p_2)$$

$\mathbf{OR}^*$ Bids

- The idea is to express `XOR` with `OR`.
- This can be achieved by using dummy items.
- Dummy items don't have any values but can constrain bids to represent `XOR`.

$$(S_1, p_1)\,\texttt{XOR}\,(S_2, p_2)$$

Can be shown as:

$$(S_1 \cup \{dummy\}, p_1)\,\texttt{OR}\,(S_2 \cup \{dummy\}, p_2)$$

- This language can simulate all bidding languages discussed so far.

OR* Bids Cont'd

$\mathbf{OR}^*$ Bids Cont'd

- It can be shown that any `OR/XOR` bid with s clause can be converted to an $\mathbf{OR}^*$ bid with s clause and s^2 dummy bids.

$\mathbf{OR}^*$ Bids Cont'd

- It can be shown that any `OR/XOR` bid with s clause can be converted to an $\mathbf{OR}^*$ bid with s clause and s^2 dummy bids.

- This can be observed by considering that there are at most $\binom{s}{2}$ mutually exclusive bid pairs.

$\mathbf{OR}^*$ Bids Cont'd

- It can be shown that any `OR`/`XOR` bid with s clause can be converted to an $\mathbf{OR}^*$ bid with s clause and s^2 dummy bids.

- This can be observed by considering that there are at most $\binom{s}{2}$ mutually exclusive bid pairs.

- The good thing is that we can use the systems compatible with `OR` bids to work with $\mathbf{OR}^*$.

Extensions to Bidding Languages: Logical Languages

Extensions to Bidding Languages: Logical Languages

- Natural extension of `OR` and `XOR` would be a bidding language that supports more logical operations.

Extensions to Bidding Languages: Logical Languages

- Natural extension of `OR` and `XOR` would be a bidding language that supports more logical operations.

- In [HB00], authors use `CNF`-like formulas (`AND-of-ORs`) to express bids.

Extensions to Bidding Languages: Logical Languages

- Natural extension of `OR` and `XOR` would be a bidding language that supports more logical operations.

- In [HB00], authors use `CNF`-like formulas (`AND-of-ORs`) to express bids.

- A more general approach would use all possible logical connectors.

Extensions to Bidding Languages: Logical Languages Cont'd

Extensions to Bidding Languages: Logical Languages Cont'd

- Another approach,in [ZBS03], uses AND on valuation function rather than on bids themselves. Authors call such ANDs, ALL.

$$v = v_1\,\mathtt{ALL}\,v_2$$
$$\begin{cases} v(S) = 0, & \text{if}\,(v_1(S) = 0)\,\text{or}\,\,(v_1(S) = 0) \\ v(S) = v_1(S) + v_2(S), & \text{otherwise} \end{cases}$$

Extensions to Bidding Languages: Logical Languages Cont'd

- Another approach,in [ZBS03], uses AND on valuation function rather than on bids themselves. Authors call such ANDs, ALL.

$$v = v_1 \, \mathtt{ALL} \, v_2$$

$$\begin{cases} v(S) = 0, & \text{if} \, (v_1(S) = 0) \, \text{or} \, (v_1(S) = 0) \\ v(S) = v_1(S) + v_2(S), & \text{otherwise} \end{cases}$$

- They use ALL along with SUM and MAX operators for expressing bids.

Extensions to Bidding Languages: Logical Languages Cont'd

- Another approach,in [ZBS03], uses `AND` on valuation function rather than on bids themselves. Authors call such `AND`s, `ALL`.

$$v = v_1 \,\texttt{ALL}\, v_2$$
$$\begin{cases} v(S) = 0, & \text{if}\,(v_1(S) = 0)\,\text{or}\,(v_1(S) = 0) \\ v(S) = v_1(S) + v_2(S), & \text{otherwise} \end{cases}$$

- They use `ALL` along with `SUM` and `MAX` operators for expressing bids.

- This approach can solve some special cases of *preference elicitation* in polynomial time.

Extensions to Bidding Languages: k-OR

Extensions to Bidding Languages: k-OR

- k-OR is the generalization of OR and XOR.

Extensions to Bidding Languages: k-OR

- k-OR is the generalization of OR and XOR.

- An evaluation on k-OR lets at most k bids to be true.

$$v = OR_k(v_1 \ldots v_t)$$
$$v(S) = \mathtt{max}_{S_1 \ldots S_k} \sum_{j=1}^{k} v_{i_j}(S_j)$$

$S_1 \ldots S_k$ form a partition of the items and $i_1 < i_2 \ldots < i_k$.

Extensions to Bidding Languages: associated prices

Extensions to Bidding Languages: associated prices

- Instead of specifying the bids' values explicitly the bidder factors out the base price (*associated prices*).

Extensions to Bidding Languages: associated prices

- Instead of specifying the bids' values explicitly the bidder factors out the base price (*associated prices*).

- Let item $\mathtt{A}$'s value to be 101 and item $\mathtt{B}$'s to be, 102. They can be considered to provide at least benefit of 100 and thus can be shown as:

$$(\,[(\mathtt{A},1)\,\mathtt{OR}\,(\mathtt{B},2)],\;100).$$

Special Cases: Symmetric Valuations

Special Cases: Symmetric Valuations

- Symmetric valuations often only deal with the number of items won.

Special Cases: Symmetric Valuations

- Symmetric valuations often only deal with the number of items won.
- The information for valuation $v_1, v_2, \ldots, v_m$ can be stored by a simple vector.

Special Cases: Symmetric Valuations

- Symmetric valuations often only deal with the number of items won.
- The information for valuation $v_1, v_2, \ldots, v_m$ can be stored by a simple vector.
- Alternatively, it is possible to store marginal values, $p_i = v_i - v_{i-1}$.

Special Cases: Symmetric Valuations

- Symmetric valuations often only deal with the number of items won.
- The information for valuation $v_1, v_2, \ldots, v_m$ can be stored by a simple vector.
- Alternatively, it is possible to store marginal values, $p_i = v_i - v_{i-1}$.
- Also, it is possible to model the information by a *demand curve d*.

Special Cases: Network Valuations

Special Cases: Network Valuations

- Often, the items for sale are network resources. The resources can be considered as edges of a graph where nodes represent locations of the network.

Special Cases: Network Valuations

- Often, the items for sale are network resources. The resources can be considered as edges of a graph where nodes represent locations of the network.

- Consider, as an example, the case where a bidder is interested to transfer information from node s to t. The bid then should consist of (s, t) and the proposed price p for network resource(s) for information transformation between s and t.

Complexity of Bidding Languages

Complexity of Bidding Languages

Complexity of bidding languages can be studied in three areas:

Complexity of Bidding Languages

Complexity of bidding languages can be studied in three areas:

- **Expressions:** How efficient different bidding languages can be translated.

Complexity of Bidding Languages

Complexity of bidding languages can be studied in three areas:

- **Expressions:** How efficient different bidding languages can be translated.
- **Winner Determination:** In general, it is independent of the bidding language and leads to NP-Complete problems.

Complexity of Bidding Languages

Complexity of bidding languages can be studied in three areas:

- **Expressions:** How efficient different bidding languages can be translated.
- **Winner Determination:** In general, it is independent of the bidding language and leads to NP-Complete problems.
- **Evaluation:** It is basically about how we can extract information from bids.

Complexity of Evaluation

Complexity of Evaluation

Two basic types of evaluation, are:

- **value query:** Given a set S what is $v(S)$?

Complexity of Evaluation

Two basic types of evaluation, are:

- **value query:** Given a set S what is $v(S)$?
- **demand query:**Given a set of items $\{p_i\}$, find the set that maximizes $v(S) - \sum_{i \in S} p_i$.

Complexity of Evaluation

Two basic types of evaluation, are:

- **value query:** Given a set S what is $v(S)$?
- **demand query:**Given a set of items $\{p_i\}$, find the set that maximizes $v(S) - \sum_{i \in S} p_i$.

A value query can be reduced to a demand query via polynomial-time Turing reduction.

Complexity of Evaluation

Two basic types of evaluation, are:

- **value query:** Given a set S what is $v(S)$?
- **demand query:**Given a set of items $\{p_i\}$, find the set that maximizes $v(S) - \sum_{i \in S} p_i$.

A value query can be reduced to a demand query via polynomial-time Turing reduction.

While, for Simple and `XOR` bids, it is possible to provide efficient algorithm for evaluation, for other bidding languages, evaluation usually leads to the problem of item allocation which is in general NP-Complete.

Complete Bidding Languages

Complete Bidding Languages

- The idea is to design a language that is capable of simulating all bidding languages.

Complete Bidding Languages

- The idea is to design a language that is capable of simulating all bidding languages.

- An idea is to submit program as valuations for the bids.

Complete Bidding Languages

- The idea is to design a language that is capable of simulating all bidding languages.
- An idea is to submit program as valuations for the bids.
- The challenge is then what should such *programs* provide.

Complete Bidding Languages

- The idea is to design a language that is capable of simulating all bidding languages.
- An idea is to submit program as valuations for the bids.
- The challenge is then what should such *programs* provide.
- It follows that such *program valuations* still hold the NP-Completeness property of other bidding languages.

Conclusion

Conclusion

- Differen Bidding Languages provide different levels of expressivity.
- Regardless of the type of bidding language, we are dealing with NP-Complete problems of allocation and winner determination.
- In this way, it seems to me that expressivity should be the main concern for the bidding languages.
- While there seems to exist a hierarchy of languages based on their expressiveness, there is no formal specification of ideal bidding language.

Reference

Literatur

[HB00] Holger H. Hoos and Craig Boutilier. Solving combinatorial auctions using stochastic local search. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 22–29. AAAI Press / The MIT Press, 2000.

[LOS99] Daniel J. Lehmann, Liaden Ita O'Callaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. In *ACM Conference on Electronic Commerce*, pages 96–102, 1999.

[Nis04] Noam Nisan. *Bidding Languages*. 2004.

[ZBS03] Martin A. Zinkevich, Avrim Blum, and Tuomas Sandholm. On polynomial-time preference elicitation with value queries. In *Proceedings of the 4th ACM conference on Electronic commerce*, pages 176–185. ACM Press, 2003.