Computation and Economics - Spring 2012
Section: Mechanism Design and Combinatorial Auctions

Professor Sven Seuken
Department of Informatics, University of Zurich

1 Mechanism Design: Impossibility Results

1.1 Boarda Scoring Rule

Consider the election of a political representative out of m candidates (alternatives) by n voters (agents). Intuitively, the *Borda scoring rule* assigns a weight to each position in the ranking of each agent, then takes the sum of the weights a candidate recieves. Formally:

- Collect preference orderings from all agents
- Define the weight of being ranked in k^{th} position by agent i as $w_{i,k} := m - k$. (Note that the weight may depend on the agent i, but in this definition it does not.)
- Define an indicator that is 1 if candidate a holds k^{th} position in the ranking of agent i, i.e.

$$x_i(a,k) := \begin{cases} 1, & \text{if } a \text{ is } i'\text{s } k^{th}\text{choice,} \\ 0, & \text{else.} \end{cases}$$

- For each candidate a compute the sum of weights

$$X(a) = \sum_{i=1}^{n} \left(\sum_{k=1}^{m} w_{i,k} x_i(a,k) \right).$$

  For the weights defined above, this means: for each time the candidate is ranked first, he receives $m-1$ points, $m-2$ for each second ranking, and so on

- Select the candidate a with the highest score $X(a)$, break ties at random

1.2 In-class exercise

1. What will be the outcome if the reports are as follows:

Agent	Report
1:	$a \succ b \succ c$
2:	$b \succ a \succ c$
3:	$a \succ c \succ b$
4:	$c \succ b \succ a$

2. What is a useful manipulation for agent 4?

2 Mechanism Design: YouTube Game

Consider the following setting: Each of n agents can produce a short movie. Producing a movie will cost him his production cost c_i, which is private information to agent i. If a movie is produced, it yields 1 unit of utility to all other agents.

In a system with n agents, the production of a movie by i will thus increase welfare by $s_i := (n-1) - c_i$. The welfare maximising strategy would ask all agents with $s_i > 0$ to produce their movie. The overall utility would be

$$\sum_{i=1}^{n} \max(s_i, 0) = \sum_{i=1}^{n} (n-1-c_i) \underbrace{\mathbf{1}_{\{c_i<n-1\}}}_{=:x_i^*}.$$

Here x is the allocation, which is 1 if agent i produces a movie and 0 otherwise.

2.1 In-class exercise: Impossibility

Agents may have quasi-linear utility functions. If we can show that the domain contains the simple exchange setting, then the impossibility results HGL und GS would apply in this context. In general impossibility results hold on larger domains if they are proven for a smaller domain.

2.2 In-class exercise: VCG

Let's try to use the VCG mechanism and find out how well it does in this domain.

The private information is just the capacities c_i, since the utility each movie provides to the other agents is common knowledge. However, VCG requires a valuation function $\hat{v}_i$ as input. The valuation function of agent i is given by

$$v_i(x) = \left(\sum_{j \neq i} x_j \right) - x_i c_i.$$

Here the agent receives 1 unit of utility from every movie produced by other agents and a negative utility of c_i from producing his own movie, if asked.

Based on the reported valuation function $\hat{v}$, the mechanism determines the allocation $x^*(\hat{v})$ and the payments $t_i(\hat{v})$.

We have already discovered that the efficient allocation is $x_i^*(\hat{v}) = \mathbf{1}_{\{\hat{c}_i<n-1\}}$, where $\hat{c}_i$ is the production cost implied by the reported valuation function. VCG will select this allocation.

The payment to be made by the agent is the difference in utility to all other agents that his presence in the game causes. If agent i was not present, the mechanism would ask all agents to produce a video, if their c_i is less than $n-2$. Define the reduced allocation x^{-i} when a agent i is not present by

$$x_j^{-i} := \mathbf{1}_{\{\hat{c}_j<n-2\}}$$

for all $j \neq i$. Then

$$\begin{aligned}
t_i(\hat{v}) &= \sum_{j\neq i} \hat{v}_j(x^{-i}) - \sum_{j\neq i} \hat{v}_j(x^*) \\
&= \sum_{j\neq i} \left(\sum_{k\neq i,j} x_k^{-i} - \hat{c}_j x_j^{-i} \right) - \sum_{j\neq i} \left(\sum_{k\neq j} x_k^* - \hat{c}_j x_j^* \right) \\
&= \sum_{j\neq i} \left[\sum_{k\neq i,j} (x_k^{-i} - x_k^*) - x_i^* - \hat{c}_j (x_j^{-i} - x_j^*) \right] \\
&= -(n-1)\mathbf{1}_{\{\hat{c}_i < n-1\}} - \sum_{j\neq i} \#\{k \neq i, j : \hat{c}_k = n-2\} + \sum_{j\neq i} \hat{c}_j (x_j^{-i} - x_j^*)
\end{aligned}$$

This means the agent will recieve a payment consisting of three components:

- If the production of a movie by i is profitable, i will produce it and cause an increase in utility to all $n-1$ other agents by 1.
- Without i present, the agents with cost $c_j = n-2$ would not produce their movie. If i joins, these agents will produce their movies, resulting in a benefit to all other agents.
- The agents with $c_j = n-2$ incur the cost of production.

In particular the mechanism runs a deficit: Payments from agents to the mechanism are never positive. Additionally, if a movie is produced, the agent producing it receives a payment, leading to a loss for the mechanism. This is because the no-positive contributions property is violated: If agent i produces a movie, removing that agent will result in a net reduction in welfare by at least $n-1-c_i > 0$. This is a strict reduction, since the movie was selected for production.

2.3 In-class exercise

Let's do an example: If 3 agents have cost $c_1 = 0, c_2 = 1, c_3 = 3$, what would be the outcome of the VCG mechanism? The allocation is $x^* = (1,1,0)$, and with one agent removed, this becomes $x^{-1} = (-,0,0), x^{-2} = (1,-,0), c^{-3} = (1,0,-)$. Then the payments are as follows:

$$t_1 = -2, t_2 = -2, t_3 = 0.$$

You can see that the mechanism ends up paying for all utility produced by the agents.

3 Combinatorial Auctions: Problem Formulation

If agents have super-additive valuation functions, this means that the value of a bundle may be greater than the sum of the values of the individual components. The intuition behind this is "synergies": If you own a PC you can store pictures on your harddrive (worth 5). If you own a printer (but no PC), you can use it for decoration (worth very little). But if you have both, you can print the stored images and decorate your living room with these instead (worth 20).

Formally, this means that for any 2 disjoint subsets $A, B \in \mathcal{P}(\mathcal{O})$ of the item space, we have

$$v_i(A) + v_i(B) \leq v_i(A \cup B).$$

Such valuation functions can be expressed using the OR-language.

To find out what the value for a particular collection $C \in \mathcal{P}(\mathcal{O})$ of items is for the bidder, we need to check what combinations of valuable collections we can fit into C. Then we take the combination with the highest sum of subset values and take this sum as the value for C. As an extreme case, an atomic bid may refer to C directly, in which case the value is exactly that bid.

Formally

$$v_i(C) = \max \left\{ \sum_{l=1}^{k} v_i(S_l) \;\middle|\; \begin{array}{l} S_1, \dots, S_k \in \mathcal{P}(\mathcal{O}), \\ \forall l \in \{1, \dots, k\} : S_l \subseteq C, \\ \forall l, m \in \{1, \dots, k\} : S_l \cap S_m = \emptyset, \\ \forall l \in \{1, \dots, k\} : \exists (S_l, v_i(S_l)) \text{atomic bid} \end{array} \right\}.$$

3.1 In-class exercise

Suppose we are interested in the valuation for {A,B,C,D,E,F} the bids

$$(\{D\}, 6) OR (\{A, B\}, 3) OR (\{B, C\}, 4) OR (\{D, E, F\}, 9) OR (\{C, D, E, F, G\}, 11).$$

The possible combinations of subsets are:

{D}	{A,B}	{B,C}	{D,E,F}	{C,D,E,F,G}	Valuation
1	1	0	0	0	9
1	0	1	0	0	10
0	1	0	1	0	12
0	0	1	1	0	13

Hence, the valuation of {A,B,C,D,E,F} would be 13.

3.2 In-class exercise

The mechanisms' allocation problem is to determine that best possible allocation from the bids. Suppose, the mechanism recieved bids in the OR language from n bidders and wants to maximize overall value. Hence the mechanism has to choose a consistent set of atomic bids such that sum of all values is maximal. We want to write this as an integer program.

First some notation: We denote by *item* any elementary good, which can not be split. By a *collection* we mean a set of items.

For each agent we define the set S_i, that contains the collections for which the agent submitted an atomic bid. Hence

$$S_i = \{A | i \text{ submitted an atomic bid } (A, v_A)\}.$$

This is used to define the overall set of collections for which atomic bids by any bidder exist, i.e.

$$S = \bigcup_{i=1}^{n} S_i = \{A | \exists i \in \{1, \dots, n\}, \text{ such that } i \text{ submitted an atomic bid } (A, v_A)\}.$$

For all agents $i \in \{1, \ldots, n\}$ and all collections of items $A \in S$ define

$$v_i(A) := \begin{cases} v_A, & (A, v_A) \text{ is an atomic bid from agent } i \\ 0, & \text{else.} \end{cases}$$

Then the integer program should maximize the sum of the values to all agents from their respective allocation, hence

$$\max_x \sum_{i=1}^{n} \sum_{A \in S} x(i, A) v_i(A), \tag{Objective}$$

where x is the allocation, i.e. the mechanism can change the values of x and only these to maximize the outcome.

Of course not every allocation x is feasible, so we need to set the right boundary conditions.

The tricky constraint is to ensure that no item is allocated twice. This is equivalent to saying that out of any two overlapping collections, at most one is allocated. Formally we get

$$\sum_{i=1}^{n} x(i, A) + \sum_{i=1}^{n} x(i, B) \leq 1 \text{ for all } A, B \in S \text{ with } A \neq B \text{ and } A \cap B \neq \emptyset. \tag{Disjoint}$$

Also we have to ensure that no item is split, so we get

$$x(i, A) \in \{0, 1\} \text{ for all } i \in \{1, \ldots, n\}, A \in S. \tag{Integrality}$$

Now we can interpret the x: $x(i, A) = 1$ indicates that we give the collection A to agent i. However it is possible the an agent is allocated multiple disjoint collections. The IP finds the combination of atomic bids that maximizes overall value. This is consistent with the OR bidding language, where the value of a collection is determined by the sum of the values of subsets for which atomic bids exist. For example, if there is only one agent and the agent submitted the bids $(\{a, b\}, 3), (\{c\}, 3), (\{b\}, 2), (\{a, c\}, 5)$, the IR would allocate $\{b\}$ and $\{a, c\}$ because the value is 7.