Algorithmic Game Theory	April 13, 2016

Lecture 7

Lecturer: Michal Feldman, Scribe: Oron Laor, Yotam Frank, Sofia Rovinsky, Noam Iluz

Combinatorial Auctions

We denote M as the set of different products that are up for sale, and m as the size of M. There are n (strategic) players. Each player i has a valuation function: $v_i : 2^m \to \mathbb{R}^+$.

Standard Assumptions: Normalization: $v_i(\emptyset) = 0$; Monotony: $v_i(S) \leq v_i(T)$ for all $S \subseteq T \subseteq M$.

Allocation: $x = (x_1, x_2, \ldots, x_n)$ where: $x_i \subseteq M$ for all i, $x_i \cap x_j = \emptyset$ for all $j \neq i$, and $\bigcup_i x_i = M$.

Goal: Finding an allocation x that Maximizes the Social Welfare (SW), defined as: $\sum_{i=1}^{n} v_i(x_i)$.

To achieve our goal, we would have been happy to use VCG, but we might encounter, among other issues, a complexity problem. For example, we would get 2^m values from each player.

The 3 main issues in AGT

1. **Representation:** we want to be able to represent all components of a game, in an efficient way.

2. **Strategic:** we want to encourage the players to tell the truth.

3. **Algorithmic:** we want to be able to describe how to maximize SW (efficiently!).

We will now look at a case in which all those 3 issues work nicely (**with** VCG).

1 Unit Demand

We denote M as the set of different products that are up for sale, and m as the size of M. There are n (strategic) players. The valuation function for each player is as follows: Each player i has $v_{i1}, v_{i2}, \ldots, v_{im}$ values for the products.

For each $S \subset M$,and for each player i we'll denote: $v_i(S) = \max_{j \in S}\{v_{ij}\}$.

A Reminder of the VCG Mechanism

The mechanism gets $\{v_i\}_i$, and then for S - an allocation that maximizes SW, and for $S' = (S'_1, \ldots, S'_n)$:

$P_i = \max'_S \sum_{j \neq i} v_i(S'_i) - \sum_{j \neq i} v_j(S'_j)$.

So, here:

1. **Representation** is not problematic. Specifically, each player submits only m values, not 2^m.

2. No **Strategic** problem as well.

3. What about the **Algorithmic** issue? this is in fact a **maximal matching** problem (!), to which we know an algorithm that runs in polynomial time.

2 Single-Minded Bidders

- Each player i will report: (S_i^*, v_i^*), while $S_i^* \subseteq M$ is a subset of products the player i desires.
- For all $S \subseteq M$: $v_i(S) = \begin{cases} v_i^* \ S_i \subseteq S, \\ 0 \ otherwise \end{cases}$.

In words: If a player gets a package that contains the subset of products she desires (S_I^*), then she pays (v_i^*). Otherwise - she pays nothing (since the player is only interested in one item). Note that **representation** still isn't a problem.

Claim 1 *Maximizing SW for Single-Minded Bidders is a (very) hard problem: Given input* (S_i^*, v_i^*) *and an integer k, determining whether it's possible to achieve* $k \leq SW$ *is NP-Hard.*

Proof: We will prove the above by a reduction from IS (Independent Set). Namely, Given a graph $G(V, E)$, and an integer k, we show that a solution for achieving $k \leq SW$ gives us a solution for IS:

For each vertex in the graph we create a player $i \in V$. We define the player's package to be: $S_i^* = \{(i, j) \in E\}$. Now we can notice that *an allocation S is valid* **iff** *the set of "winners":* $\{i : S_i^* \subset S_i\}$ *is an Independent Set in the original graph.* ■

From the proof above, we can say: $k \leq SW \Leftrightarrow IS \geq k$. Because of that (elaboration omitted), we can say that we can't approximate the solution better than $\sqrt{m}$.

Claim 2 *The exist a* ***truthful, polynomial*** *mechanism, that gives an approximation rate of* $\sqrt{m}$.

The Mechanism

- Each player i reports: (S_i^*, v_i^*).
- The players are sorted by: $\frac{v_1^*}{\sqrt{|S_1^*|}} \geq \frac{v_2^*}{\sqrt{|S_2^*|}} \ldots, \frac{v_n^*}{\sqrt{|S_n^*|}}$.
- We begin with a greedy allocation: $w \leftarrow \emptyset$ for all $i = 1, \ldots, n$.
- If $\emptyset = S_i^* \cap (\bigcup_{j \in w} S_j^*)$, then $w \leftarrow w \cup \{i\}$. (If the products the player wants do not intersect with already-given products - we simply add her to the allocation).
- **Payments:** Player i pays the lowest value she could have reported and still win: $P_i = v_j^* \frac{\sqrt{|S_i^*|}}{\sqrt{|S_j^*|}}$, such that j is the smallest index for which: $S_i^* \cap S_j^* \neq \emptyset$ and for all $k \leq j, k \neq i$ we have $S_k^* \cap S_j^* = \emptyset$. In other words: If you don't stand in anybody's way - you can get your products for free. Otherwise, she will have to pay according to player j described, the one that she stood on his way.

The next few steps will eventually show that:

1. The algorithm for allocation is polynomial.
2. The algorithm gives an approximation rate of $\sqrt{m}$.
3. The mechanism is **truthful**.

Claim 3 *The mechanism is monotone i.e if a player won by bidding* (S_i^*, v_i^*) *he will still win by bidding* (S_i', v_i') *s.t* $v_i' \geq v_i^*$, $S_i' \subseteq S_i^*$.

Proof: Whether a player i wins S_i^* depends solely on his position in our sorting. The larger $\frac{v_i'}{\sqrt{|S_i'|}}$ the sooner our algorithm will encounter player i the sooner player i is encountered the more he wins. ■

Claim 4 *The payment* p_i *is the critical value i.e the minimal value* v_i' *for which* (S_i^*, v_i*) *still gains* S_i^*.

Proof: Player i wins as long as it shows ahead of j in the sorting i.e: $\frac{v_i^*}{\sqrt{|S_i^*|}} \geq \frac{v_j^*}{\sqrt{|S_j^*|}}$ and by simply dividing both sides of the inequality we get $v_i^* \geq v_j^* \frac{\sqrt{|S_i^*|}}{\sqrt{|S_j^*|}}$. The right side of the inequality is exactly equal to the payment p_i. ■

Theorem 5 *There exists a mechanism, both truthful and polynomial, which approximate the optimal social welfare within multiplicative factor of* $\sqrt{m}$ *i.e* $\frac{OPT}{ALG} \leq \sqrt{m}$ *.*

Proof: The algorithm is obviously polynomial so we begin by showing truthfulness. First note that for a lie (S_i', v_i') to be profitable for player i then necessarily $S_i^* \subset S_i'$. From monotonicity if (S_i^*, v_i^*) wins then so does (S_i^*, v_i') without augmenting player i payment. **Conclusion:** $(S_i^*, v_i^*) \succ (S_i', v_i')$ payment-wise. It's enough to show $(S_i^*, v_i^*) \succ (S_i^*, v_i')$. Indeed let a lie v_i'.

- If v_i' didn't win then the value for player i is 0 and the utility for player i is at most 0.
- If v_i' wins then the payment still is $v_j^* \frac{\sqrt{|S_i^*|}}{\sqrt{|S_j^*|}}$ the same payment.

Therefore we showed that in any case it is not profitable to lie. ■

Claim 6 *The mechanism achieves approx. ratio* $\sqrt{m}$*.*

Proof: First we show $\sum_{i \in w} v_i^* \geq \frac{1}{\sqrt{m}} \sum_{i \in OPT} v_i^*$. Denote $OPT_i = \{j \in OPT : j \geq i, S_i^* \cap S_j^* \neq \emptyset\}$. Notice if player i wins S_i^* both in ALG and in OPT i.e $i \in OPT \cap w$ then $i \in OPT_i$. We'll show:

1. For any player i: $OPT \subseteq \bigcup_{i \in w} OPT_i$.
2. For any $i \in w$: $\sum_{j \in OPT_i} v_j^* \leq \sqrt{m} v_i^*$.

(1) If $i \in w$ then obviously $S_i^* \cap S_j^* \neq \emptyset$ and $i \in OPT_i$. If $i \notin w$ then $i \in OPT_j$ s.t $j \in w$ and $j < i$ and $S_i^* \cap S_j^* \neq \emptyset$. Necessarily there exists a j like that otherwise we would have $i \in w$ which is a contradiction.
(2) First for any $j \in OPT_i$ we have:

$$v_j^* \leq v_i^* \frac{\sqrt{|S_i^*|}}{\sqrt{|S_j^*|}} \Rightarrow \sum_{j \in OPT_i} v_j^* \leq \frac{v_i^*}{\sqrt{|S_i^*|}} \sum_{j \in OPT_i} \sqrt{|S_j^*|}. \tag{1}$$

And by Cauchy-Schwartz inequality:

$$\sum_{j \in OPT_i} v_j^* \leq \frac{v_i^*}{\sqrt{|S_i^*|}} \sqrt{|OPT_i|} \sqrt{\sum_{j \in OPT_i} |S_j^*|}. \tag{2}$$

Consider for any $i \in w$ there exists $|OPT_i| \leq |S_i^*|$. Why? Because for any player $j \in OPT_i$ his package S_j^* intersects S_i^* and does not intersect any other package in OPT_i meaning we can assign any $j \in OPT_i$ a unique element $s \in S_i^* \cap S_j^*$ or in other words there is one-to-one

$f : OPT_i \to S_i^*$. And therefore we have:

$$\sum_{j \in OPT_i} v_j^* \le \frac{v_i^*}{\sqrt{|OPT_i|}} \sqrt{|S_i^*|} \sqrt{\sum_j |S_j^*|} \le \frac{v_i^*}{\sqrt{|OPT_i|}} \sqrt{|OPT_i|} \sqrt{\sum_j |S_j^*|} \le v_i^* \sqrt{\sum_j |S_j^*|} \le v_i^* \sqrt{m}. \quad (3)$$

Finally we have:

$$\sum_{j \in OPT} v_j^* \le \sum_{i \in w} \sum_{j \in OPT_i} v_j^* \le \sqrt{m} \sum_{i \in w} v_i^*. \quad (4)$$

Which completes the proof. ■

3 Multi Unit Auctions

Consider m identical products and n players. For each player i define a valuation function: $v_i : \{0, 1, \ldots, m\} \to \mathbb{R}^+$, where $v_i(k)$ is the value of k identical products for player i.

Standard Assumptions: Normalization: $v_i(0) = 0$; Monotony: $v_i(k) \le v_i(k+1)$ for all k.

Allocation: $x = (x_1, x_2, \ldots, x_n)$ where: $\sum_{i=1}^n x_i \le m$ and each x_i is the number of elements allocated to player i.

Goal: Finding an allocation x that Maximizes the SW, defined as: $\sum_{i=1}^n v_i(x_i)$.

Recall that we are facing 3 challenges:

1. Representation.
2. Algorithm.
3. Strategy.

We will focus at number 1.

3.1 Representation

We will ignore the real number representation issue. For a large m, we would like to have a compact representation of the valuations. However, in the general case(i.e - valuations represents as a real number), it is impossible to compress all the valuations.
There are 2 possible approaches to this problem:

1. Bidding languages.
2. Black Box - Query access model.

Bidding Languages

Using bidding languages it is possible to represent some of the valuations in a compact way. We will now describe several bidding languages, and the syntax and the semantics of each of them.

1. **Single-minded**

   Syntax: For each player i we have a pair (k_i^*, w_i^*).
   Semantics: $v_i = \begin{cases} w_i^* & \text{if } k \geq k^* \\ 0 & \text{if } otherwise \end{cases}$.

2. **Step-function**

   Syntax: For each player i we have list of pairs $(k_{i1}, w_{i1}), (k_{i2}, w_{i2}), \ldots, (k_{it}, w_{it})$.
   Semantics: $v_i(k) = w_{ij}$ for max j s.t $k \geq k_{ij}$.

   Example:

   Syntax: $(2, 7), (5, 23)$.
   Semantics: 2 is the minimal k that receives value, hence: $V(0) = v(1) = 0$.
   For $2 \leq k < 5$ we have: $v(2) = v(3) = v(4) = 7$.
   For $5 \leq k$ we have: $v(5) = v(6) = \ldots = 23$.

3. **Piecewise linear (PWL)**

   Syntax: For each player i we have a sequence of pairs $(k_{i1}, p_{i1}), (k_{i2}, p_{i2}), \ldots, (k_{it}, p_{it})$.
   Semantics: The value of player i is defined by the marginal values which are represented by the given sequence of pairs. Meaning, for each $1 \leq l \leq k$ define $u_{il} = p_{ij}$ for min j s.t $l \leq k_{ij}$. Then define: $v_i(k) = \sum_{l=1}^{k} u_l$.

   Example:

   Syntax: $(2, 7), (5, 23)$.
   Semantics:
   $v(0) = 0$.
   $v(1) = 7$.
   $v(2) = 14$.
   $v(3) = 37$.
   $v(4) = 60$.
   $v(5) = 83$.

We want to examine the expressiveness of the languages:

1. First we notice that the step language includes the single-minded language: a single-minded valuation (k_i^*, w_i^*) is also a step valuation with a single pair.

2. We'll examine the relation between step-function and PWL:

   - We will convert a step valuation $(k_{i1}, w_{i1}), (k_{i2}, w_{i2}), \ldots, (k_{it}, w_{it})$ to PWL valuation as follows:
     For each step (k_{ij}, w_{ij}) we will define: $(k_{ij}, w_{ij} - w_{ij-1}), (k_{ij+1}, 0)$.
   - Converting a PWL valuation to a step valuation may increase substantially the number of values required for representation. For example, the PWL valuation $(m, 1)$ requires m step values in a step function: (k, k) for each k.

3.2 Black Box

In this approach, we have an interface we can use to query a "black box" regarding the valuations.

We'll consider our results as "good" results in one of the two following cases:

- Positive results for weak queries.
- Negative results for strong queries.

Query types:

- **Value query**: given k, want to find $v(k)$. This is a weak query.
- **Demand query**: given a sequence of product prices: $p_1, p_2, \ldots, p_m$, which subset $s \in S$ maximizes the utility of a specific player, defined as: $v(s) - \sum_{j \in S} p_j$. This is a strong query.