1 Introduction

We discussed Combinatorial Auctions in the simple case of single minded bidders in order to avoid dealing with exponential input. The problem of maximizing \( \sum_{i \in w} v_i \), where \( w \) is the group of the winners, is NP-hard. We showed a mechanism which gives approximation of \( \sqrt{m} \), which is the best approximation possible. Let’s remember that the mechanism ordered the bidders by their \( \frac{v_i}{\sqrt{|s_i|}} \) and made player \( i \) pay \( \frac{v_j \times \sqrt{|s_i|}}{\sqrt{|s_j|}} \) where \( j \) is the first player who didn’t win only because of player \( i \). We will continue discussing this mechanism, and then discuss the general problem of combinatorial auctions.

2 More about the single minded bidders case

**Theorem 1** The mechanism is incentive compatible.

We will use the following lemma:

**Lemma 2** A single minded mechanism is incentive compatible if and only if the following conditions hold:

- The mechanism is monotonic, meaning:
  \[ \forall v_{-i} : v'_i > v_i \rightarrow i \text{ wins saying } v_i, \text{ he’ll also win saying } v'_i \]
  \[ \forall s_{-i} : s'_i \subseteq s_i \rightarrow i \text{ wins saying } s_i, \text{ he’ll also win saying } s'_i \]

- If player \( i \) wins he pays his critical value, which is the smallest \( v_i \) he can say and still win.

We will now prove the theorem using the lemma by simply verifying that the conditions of the lemma hold in the mechanism we described for the single minded bidders case.

**Proof:**
The mechanism is monotonic: If a player $i$ declares $v'_i > v_i$ he will be higher in the order by $\frac{v_i}{\sqrt{|s_i|}} (\frac{v'_i}{\sqrt{|s'_i|}} > \frac{v_i}{\sqrt{|s_i|}})$, so he has a better chance of winning. The same goes for $s'_i \subseteq s_i$. The player will also go up in the order $(\frac{v_i}{\sqrt{|s_i|}} > \frac{v_j}{\sqrt{|s_j|}})$ and $s'_i$ will be cut with less packages. All that makes his chances of winning bigger.

If player $i$ wins he pays his critical value: The critical value of player $i$ which is the only one to prevent player $j$ from winning (and $j$ is the first to be prevented) is the minimum value $v^*$ such that $\frac{v^*}{\sqrt{|s'_i|}} > \frac{v_j}{\sqrt{|s_j|}}$ meaning $v^* > \frac{v_j \times \sqrt{|s_i|}}{\sqrt{|s_j|}}$ (he’ll still win since $s_i$ has no products in common with any package in $s_{j+1} \ldots s_{i-1}$ in the order - otherwise, they wouldn’t have won), and this is exactly player $i$’s payment. ■

3 The proof of the lemma

Proof: We will show only one direction of the lemma: If the conditions hold in a mechanism, it is incentive compatible. Let’s say that player $i$, who wins now, wants to lie and say he wants $s'_i \supseteq s_i$. But it will only reduce his chances of winning, since $\frac{v_i}{\sqrt{|s'_i|}} < \frac{v_i}{\sqrt{|s_i|}}$ and there is a bigger chance to be cut with another package. He will also have to pay more if he wins: His critical value is bigger now (according to the first condition), and then he will pay more (according to the second condition).

Let’s say that player $i$ doesn’t win now. Will saying a false $s_i$ help him win? Once again, declaring a bigger package will make him lower in the order and will make his package cut with more packages, so his chances of winning will be smaller. Declaring a smaller package $s'_i \subseteq s_i$, in both cases, is irrelevant because then he will not get the products he wants.

We concluded that cheating by saying a false $s_i$ will not do any good.

What happens if player $i$, who wins now, declares a false $v_i$? If player $i$ wins, it means that $v_i > v^*$ (the critical value in order to win). If he declares more that that, he will still win and pay the same ($\frac{v_i \times \sqrt{|s_i|}}{\sqrt{|s'_i|}}$ does not depend on $v_i$). If he declares $v_i < v^*$ he might not win. If player $i$ does not win now and he declares a smaller $v_i$ he will still not win (he will be even lower in the order by $\frac{v_i}{\sqrt{|s_i|}}$). If he declares $v_i$ that is not high enough in order to win - he doesn’t gain anything. If he declares a bigger $v_i$ so that he wins, he will end up paying more than his real value for the package (because he will need a bigger value than the one of the player that made him not win, whose value is bigger than his).

We concluded that cheating by saying a false $v_i$ will not do any good.

All together, we showed that if the conditions of the lemma hold, the mechanism is incentive compatible

■
4 Back to the general case of combinatorial auctions

Let’s repeat the general problem:

Input: $v_i : 2^m \rightarrow R$

Output: maximize $\sum_i v_i(s_i)$ when $s_1 \ldots s_n$ is a partition.

Assumptions:

$v_i(\emptyset) = 0$

$S \subseteq T \rightarrow v(S) \leq v(T)$

We would like to find a way to deal with the problem of exponential input. Theoretically, we can think of two options to deal with the problem:

- Make up a language that enables to easily define the valuations of the bidders.
- Black box: when needed, each player will be asked for his value for a certain package.

We will see how formalizing the problem will help us.

Here it is:

Maximize $\sum_{X_{i,s}} v_i(s)$

S.T.

$\forall i, s : X_{i,s} \in \{0, 1\}$ (1 if player $i$ gets package $s$, and 0 otherwise)

$\forall j : \sum_{i,s:j} X_{i,s} \leq 1$ (each product $j$ was given to at most one player)

$\forall i : \sum_{s} X_{i,s} \leq 1$ (each player got at most one package)

when

$i \in \{1 \ldots n\}$ is a player,

$j \in \{1 \ldots m\}$ is a product,

$s \subseteq \{1 \ldots m\}$ is a package.

We have an Integer programming problem with $n \times 2^m$ variables. This problem is NP-hard. We want to relax it, and convert it to a linear programming problem, which is much easier to solve. Let’s define it.

Maximize $\sum_{X_{i,s}} v_i(s)$

S.T.
∀i, s : X_{i,s} ≥ 0
∀j : \sum_{i,s \ni j} X_{i,s} ≤ 1
∀i : \sum_s X_{i,s} ≤ 1

Then, of course we will get real numbers in [0..1] and the value we want to maximize may be bigger than the value we get in the Integer Programming problem.

**Example 1** Let’s say we have three bidders a, b, c and three products A, B, C.

Bidder a wants products A, B for $1.
Bidder b wants products B, C for $1.
Bidder c wants products C, A for $1.

If we don’t allow a linear solution, only one player will get the package he wants. The maximum value will be 1. However, if we allow a linear solution each product will be split to half, and each player will get half a product from each product in the package he wants. The max value will be 1.5 which is bigger than 1.

Apparently, it is possible to solve the linear program, even if we have an exponential number of variables and a polynomial number of equations (one for each player and one for each product) if we convert it to the dual problem: a variable for each equation and a constraint for each variable. Then, with a polynomial number of variables and an exponential number of equations the problem can be solved efficiently.

**Did you know?** There are some groups of problems for which, when you relax to the linear problem, you get, in fact, an integer solution: the solution to the original problem.

5 Economic point of view

The economic dogma claims that when we have a free market, prices go up and down until they reach equilibrium (An auction like we describe, where each one pays a different price, doesn’t suit this description). Let’s see what this equilibrium is.

6 Walrasian equilibrium

**Definition 3** A Walersian equilibrium is a partition \((s_1 \ldots s_n)\) and prices \(p_1 \ldots p_n\) for products \(1 \ldots n\), so that each \(i\) prefers \(s_i\) over each other package in prices \(p_1 \ldots p_m\), meaning:
∀i : ∀T : v_i(s_i) - \sum_{j \in s_i} p_j ≥ v_i(T) - \sum_{j \in T} p_j.
7 The First Welfare Theorem

**Theorem 4** If Walrasian equilibrium exits with \((s_1...s_n)\), then \((s_1...s_n)\) is an optimal partition, meaning for each other partition \((T_1...T_n)\) \[\sum_i v_i(s_i) \geq \sum_i v_i(T_i)\]

**Proof:** for some partition \((T_1...T_n) \neq (s_1...s_n)\), according to the definition of Walrasian equilibrium, we sum for all \(i\) and get:

\[
\sum_i v_i(s_i) - \sum_j p_j \geq \sum_i v_i(T_i) - \sum_j p_j
\]

\[\Rightarrow \sum_i v_i(s_i) - \sum_i \sum_j p_j \geq \sum_i v_i(T_i) - \sum_j p_j\]

\[\Rightarrow \sum_i v_i(s_i) \geq \sum_i v_i(T_i) \blacksquare\]

Now, we are going to show something stronger then the first welfare theorem.

8 The Second Welfare Theorem

**Theorem 5** If there is an optimal round solution for the linear problem, then there is a Walrasian equilibrium with the partition of this solution.

The other way is also true:

**Theorem 6** If a Walrasian equilibrium exists with \((s_1...s_n)\) then, it is even better then all linear solutions (partitions):

\[\sum_i v_i(s_i) \geq \sum_i X_i,s v_i(s)\]

**Conclusion:** The only case of a Walrasian equilibrium is when the solution of the linear program is a round solution.

In the example we’ve seen earlier, there was a gap between the round solution and the linear solution. In this case there are no prices that will create a Walrasian equilibrium.

We will now prove the second side of the Second Welfare Theorem.

**Proof:** Since a Walrasian equilibrium exists: \(\forall i : \forall s : v_i(s_i) - \sum_j p_j \geq v_i(s) - \sum_j p_j\)

We will multiply each side by \(X_{i,s}\) (the linear solution) and sum for all \(i, s\)

\[\Rightarrow \sum_i X_{i,s} * (v_i(s_i) - \sum_j p_j) \geq \sum_i X_{i,s} * (v_i(s) - \sum_j p_j)\]

\[\Rightarrow \sum_i X_{i,s} * v_i(s_i) - \sum_i X_{i,s} \sum_j p_j \geq \sum_i X_{i,s} * v_i(s) - \sum_i X_{i,s} \sum_j p_j\]

\[\Rightarrow \sum_i v_i(s_i) - \sum_i X_{i,s} \sum_j p_j \geq \sum_i X_{i,s} * v_i(s) - \sum_i X_{i,s} \sum_j p_j\]
\[ \Rightarrow \sum_i v_i(s_i) \geq \sum_{i,s} X_{i,s} * v_i(s) \]

This is it (for now) about combinatorial auctions. Next time we will discuss Iterative Combinatorial Auctions.