

# Lecture Notes: Introduction to Mechanism Design and Auctions

Berthold Vöcking

Department of Computer Science  
RWTH Aachen  
Germany

January 22, 2008

- 1 The VCG Mechanism
  - Motivating Example: Path Auctions
  - Other Examples: Procurement Auctions
  
- 2 Mechanism Design with Approximation Algorithms
  - Monotone Algorithms
  - Combinatorial Auction
  - Multi-Unit Auctions

## Motivating example: path auctions

Suppose  $n$  bidders compete for connections in a network  $G = (V, E)$  with capacities  $c : E \rightarrow \mathbb{R}_+$ .

### Bidder $i$ ...

- wants to allocate a path between a known source  $s_i$  and a known destination  $t_i$
- is single-minded: if she gets a path with a bandwidth share matching her demand  $d_i$  then her valuation is  $v_i$ , else 0.

**Objective:** Find suitable paths for a subset of bidders  $S \subseteq \{1, \dots, n\}$  such that the capacities are respected and the *social welfare*  $\sum_{i \in S} v_i$  is maximized.

Bidders are selfish! How do we convince them to reveal their demands and valuations?

### An incentive compatible mechanism ...

- computes suitable paths for a subset of the bidders, and
- charges prices to these bidders in such a way that
- truth-telling is a dominant strategy for all players.

The so-called Vickery-Clarke-Groves (VCG) mechanism is such an incentive compatible mechanism.

### The VCG mechanism ...

- collects valuations  $v'_i$  and demands  $d'_i$  from all bidders
- computes the optimal allocation  $S^*$  for the specified demands and valuations and
- charges each player  $i \in S^*$  the difference between the social welfare of the others and what it would have been without him, that is, bidder  $i \in S^*$  has to pay  $V' - (V - v'_i)$ , where
  - $V = \sum_{k \in S^*} v'_k$ ,
  - $V' = \sum_{k \in S'} v'_k$  with  $S'$  denoting the optimal solution over the bidders  $\{1, \dots, n\} \setminus \{i\}$ .

# Buying a min-cost spanning tree

- Suppose a buyer wants to buy a min-cost spanning tree in a complete graph  $G = (V, E)$  with edge costs  $c_e$ .
- Each edge is managed by a selfish agent. If edge  $e$  is part of the selected spanning tree, agent  $e$  shall receive a payment for being part of the spanning tree.

# Buying a min-cost spanning tree

## The VCG mechanism ...

- collects cost values  $c'_e$ , for all  $e \in E$
- computes the min-cost spanning tree  $T^* \subseteq E$  with respect to these values
- charges each player  $e \in T^*$  the difference between the cost of  $T^*$  without the cost for agent  $e$  and the cost of the min-cost spanning tree  $T'$  assuming edge  $e$  would not exist, that is, agent  $e \in E$  receives the payment  $C' - (C - c_e)$ , where
  - $C = \sum_{e \in T^*} c'_e$ ,
  - $C' = \sum_{e \in T'} c'_e$  with  $T'$  denoting the min-cost spanning tree if  $e$  would not exist.

## Buying a min-cost path

- Suppose a buyer wants to buy a min-cost path from a node  $s$  to a node  $t$  in a complete graph  $G = (V, E)$  with edge costs  $c_e$ .
- Each edge is managed by a selfish agent. If edge  $e$  is part of the selected path, agent  $e$  shall receive a payment for being part of the path.

# Buying a min-cost path

## The VCG mechanism ...

- collects cost values  $c'_e$ , for all  $e \in E$
- computes the min-cost path  $P^* \subseteq E$  with respect to these values
- charges each player  $e \in P^*$  the difference between the cost of  $P^*$  without the cost for agent  $e$  and the cost of the min-cost path  $P'$  assuming edge  $e$  would not exist, that is, agent  $e \in E$  receives the payment  $C' - (C - c_e)$ , where
  - $C = \sum_{e \in T^*} c'_e$ ,
  - $C' = \sum_{e \in T'} c'_e$  with  $P'$  denoting the min-cost path from  $s$  to  $t$  if  $e$  would not exist.

Let us go back to the path auction.

VCG requires the exact computation of an optimal solution for this problem. However, the underlying optimization problem is NP-hard.

Incentive compatibility is lost when VCG is combined with approximation algorithms or heuristics that do not guarantee an optimal allocation.

What can we do? - We need a different approach than VCG. We use monotonicity ...

## Monotonicity

For an algorithm  $A$  for the path auction problem, let  $S(A(d, v))$  denote the set of served bidders. We say that  $A$  is *monotone* if

$$i \in S(A((d_1, \dots, d_n), (v_1, \dots, v_n))) \Rightarrow \\ i \in S(A((d_1, \dots, d_{i-1}, d'_i, d_{i+1}, \dots, d_n), (v_1, \dots, v_{i-1}, v'_i, v_{i+1}, \dots, v_n)))$$

for any  $d'_i \leq d_i$  and  $v'_i \geq v_i$ .

## Threshold values

If an algorithm  $A$  is monotone, and all parameters except  $v_i$ , for some  $i \in \{1, \dots, n\}$ , are assumed to be fixed, then there is a threshold value  $v_i^*$  for bidder  $i$  such that

- if she bids more than  $v_i^*$  then she is part of the solution, and
- if she bids less than  $v_i^*$  then she is not part of the solution.

Similar thresholds exist also wrt demands.

**Critical value pricing:** Charge threshold valuations  $v_i^*$ .

(The critical value prices can be computed in polynomial time using a binary search.)

## Lemma: (Lehmann, O'Callaghan, Shoham)

A mechanism is incentive compatible if it is monotone and exact (i.e., no bidder gets more bandwidth than her demand) and uses critical value pricing.

### Proof:

- Lying about the valuation will not help: If I win, then I anyway pay the smallest value that will win. If the truth will make me loose, however, then I really don't want to win, since my payment will be higher than my real value.
- Lying about the demand will not help either: Increasing the demand can only lower my chances to be served. Decreasing the demand means that I'm not satisfied when I'm served.



## Combinatorial Auctions (Single-Minded Version)

In a *single-minded combinatorial auction*

- a set of  $m$  items  $U$  should to be auctioned among
- $n$  single-minded agents (*bidders*)

in a way that maximizes *social welfare*. Each bidder  $i$  declares

- $S_i$ , the set of items she is interested in, and
- $v_i$ , the maximum price she would be willing to pay.

Underlying optimization problem: Set Packing

Given  $S = (S_1, \dots, S_n)$  and  $v = (v_1, \dots, v_n)$ , find a set  $A \subseteq \{1, \dots, n\}$  that satisfies  $|\{j \in A : e \in S_j\}| \leq 1$ , for all  $e \in U$ , and maximizes  $\sum_{i \in A} v_i$ .

# Greedy Algorithm for Combinatorial Auctions

## CA-Greedy

**Input:** sets  $S_1, \dots, S_n$  and their values  $v_1, \dots, v_n$

**Output:** subset  $A$  of bidders receiving their set

01 Sort bids in non-increasing order according to the ratio

$$\frac{v_i}{\sqrt{|S_i|}}$$

02 Consider the bids in this order and assign the set  $S_i$  to bidder  $i$  if all items in  $S_i$  are still available

# Greedy Algorithm for Combinatorial Auctions

- This algorithm guarantees a  $\sqrt{m}$ -approximation for set packing.
- It is monoton and truthfull in the valuations and the sets.
- Hence, together with critical pricing it yields a truthfull mechanism.

**Exercise:** Show how to compute the critical prices.

## Def: single-minded multi-unit auctions

- Auctioneer wants to sell  $b \in \mathbb{N}$  units of a good.
- There are  $n \geq 1$  single-minded bidders.
- Bidder  $i$  is interested in buying  $a_i \in \mathbb{N}$  units of the good.
- Her valuation for getting this amount is  $v_i \in \mathbb{R}_+$ , that is, for an assignment  $x \in \mathbb{R}_+^n$ , her valuation is

$$v_i(a_i, x) = \begin{cases} v_i & \text{if } a_i \leq x_i \\ 0 & \text{otherwise} \end{cases}$$

- **Objective:** find an assignment  $x$  that maximizes the social welfare  $\sum_i v_i(a_i, x)$ .

# Underlying optimization problem

## The knapsack problem:

Given

- a knapsack of capacity  $b$  and
- $n$  objects with weights  $a_1, \dots, a_n$  and values  $v_1, \dots, v_n$
- w.l.o.g.,  $a_i \leq b$

Compute set  $S \subseteq \{1, \dots, n\}$  such that  $\sum_{i \in S} v_i$  is maximized under  $\sum_{i \in S} a_i \leq b$ .

## Monotone 2-Approximation by Mu'alem and Nisan

- Let  $A_1$  denote the greedy algorithm for KNAPSACK.
- Let  $A_2$  denote the algorithm returning the single object with highest value.
- Call  $A_1$  and  $A_2$  and output the better solution.

Unfortunately, taking the maximum over two monotone algorithms does not necessarily give a monotone algorithm. One needs an additional property ...

For simplicity, let us assume that the demands are known and bidders can only lie about their valuations.

## Bitonicity

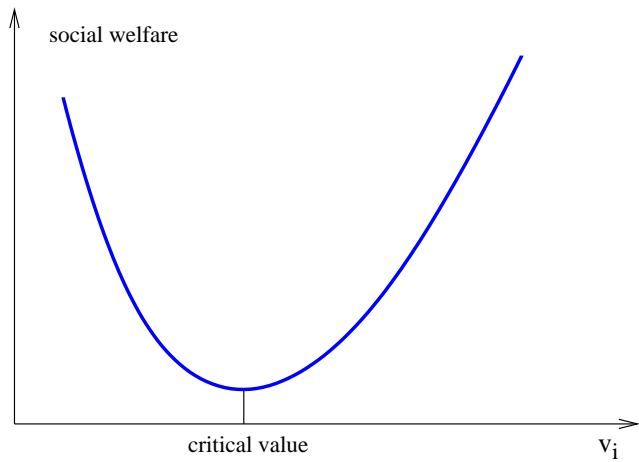
An algorithm  $A$  is called *bitonic* if it is monotone and if, for every bidder  $i$ , the social welfare of the solution computed by  $A$  is

- non-increasing in the valuation  $v_i$  for  $v_i \leq v_i^*$ , and
- non-decreasing in the valuation  $v_i$  for  $v_i \geq v_i^*$ ,

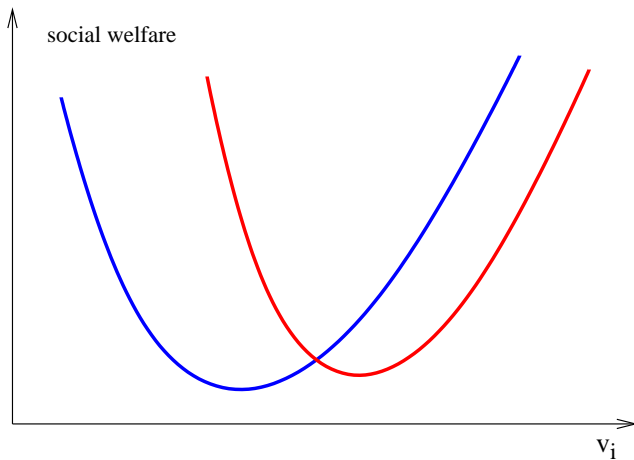
assuming that all other parameters than  $v_i$  are fixed arbitrarily.

**Lemma:** [Mu'alem and Nisan] Taking the maximum over two or more “bitonic” algorithms gives a “bitonic” and, hence, monotone algorithm.

# Bitonic Functions



# Bitonic Functions



# Bitonic Functions

