



# Foundation of Algorithmic Mechanism Design

*GI-Dagstuhl-Seminar*

*“Game-theoretic Analyses of the Internet”*

*August 30, 2004*

Reinder B. Lok

Sascha Wolf

R.Lok@KE.unimaas.nl / S.Wolf@KE.unimaas.nl



Maastricht University

# Algorithmic Mechanism Design



- Mechanism design: incentives
- Computer science: computational complexity
- Algorithmic mechanism design: both
- Nisan and Ronen (2001) “Algorithmic Mechanism Design”
- Focus: direct revelation, dominant strategy, centralized computation
- Distributed algorithmic mechanism design: decentralized computation



# Outline 1

- Introduction
- Basic definitions and assumptions
- VCG calculation in utilitarian problems
  - Utilitarian problems
  - VCG mechanisms
  - Accelerated computation of VCG-payments
  - Infeasibility of VCG

# Outline 2

---



- Alternative solution techniques for non-utilitarian problems
- Distributed algorithmic mechanism design
- Further issues



# Conflicting interests



- Set of agents  $N$
- Private information of agents:  $t^i \in T^i, i \in N$
- Set of outputs:  $O$
- Agents have preferences over outputs:  
 $v^i(t^i, o), o \in O$
- Social Choice Function:  $g(t, o)$

Conflict:  $o^*$  optimizes  $g()$   $\Rightarrow \Leftarrow o^i$  optimizes  $v^i()$



# Definitions: Mechanism



Basic structure:

1. Choose mechanism  $m = (o, p)$
2. Agents choose action  $a^i \in A^i, i \in N$
3. Output and payments are calculated:  $o(a),$   
 $p(a) = (p^1(a), \dots, p^n(a))$

Goal: Choose  $m$  such that

- Agents have dominant strategies  $a = (a^1, \dots, a^n)$
- $o(a)$  optimizes  $g(t, \cdot)$



# Definitions: Dominant strategy

- $a = (a^1, \dots, a^n)$ ,  $a \in A = A^1 \times \dots \times A^n$
- $a^{-i} = (a^1, \dots, a^{i-1}, a^{i+1}, \dots, a^n)$ ,  $a^{-i} \in A^{-i}$

## Dominant Strategy

A strategy  $a^i \in A^i$  is called dominant if for all  $\tilde{a}^i \in A^i$  and for all  $a^{-i} \in A^{-i}$  we have

$$v^i(t^i, o(a)) + p^i(a) \geq v^i(t^i, o(\tilde{a}^i, a^{-i})) + p^i(\tilde{a}^i, a^{-i})$$

# Definitions: Revelation principle

## Direct revelation mechanism:

Agents have to report their type, i.e.  $A^i = T^i$ .

## Truthful mechanism:

Truth-telling is a dominant strategy.

## Revelation principle

Suppose  $a \in A$  is a dominant strategy for  $m = (o, p)$ , and  $o(a)$  optimizes  $g(t, .)$ . Then:

$\exists$  a direct revelation mechanism  $\tilde{m} = (\tilde{o}, \tilde{p})$  such that:

- it is truthful (reporting the true types  $t \in T$ )
- $\tilde{o}(t)$  optimizes  $g(t, .)$ .

# Definitions: EFF IR BB



Desirable properties of mechanisms:

**Efficiency**  $o(a)$  maximizes  $g(t, .)$

**Individual Rationality**  $v^i(t^i, o(a)) + p^i(a) \geq 0$

**Budget-Balance**  $\sum_{i \in N} p^i(a) \leq 0$



# Definitions: Algorithms

- The output and the payments are determined with algorithms
- An **algorithm** is a precise and universally understood sequence of instructions that solve any instances of rigorously defined computational problems
- Algorithmic mechanism design: how *fast* are the algorithms that compute the mechanism

# Definitions: Complexity

- An algorithm is called *polynomial time computable* if the maximum number of basic calculations is bounded by a polynomial function of the problem size.
- Problem class  $P$  of polynomial time computable (=tractable=easy) problems
- Problem class  $NP$ -hard of problems that can not be solved by polynomial algorithms (unless ' $P = NP$ ')

# Overview



- Utilitarian problems and the VCG mechanism
  - Accelerated computation of VCG payments
  - Using approximation algorithms
- Non-utilitarian problems
  - Task scheduling
  - Deterministic and randomized algorithms
  - Verification mechanisms
- Distributed algorithmic mechanism design
  - Examples



# Utilitarian Problems



## Definition:

- mechanism design optimization problem
- select outcome  $o \in O$  maximizing

$$g(o, t) = \sum_{i=1}^n v^i(t^i, o)$$



# VCG Mechanism



**Definition:** Let  $w = (w^1, \dots, w^n)$  denote a vector of declared types. A direct revelation mechanism  $m = (o, p)$  belongs to the VCG family if:

- output  $o(w)$  maximizes  $g(o, w) = \sum_{i=1}^n v^i(w^i, o)$
- payments:  $p^i(w) = \sum_{j \neq i} v^j(w^j, o) + h^i(w^{-i})$ , where  $h^i(w^{-i})$  is arbitrary function

**Theorem:** (Groves (1973)) A VCG mechanism is truthful.

**Example:** Vickrey's second price auction

**Computational problem:**

- output determination

- payment calculation

# Accelerated computation: payments

Payment in VCG mechanisms:

$$p^i = (V - h(t^{-i})) - v^i(t^i, o)$$

Choose  $h(t^{-i}) = V^{-i}$  (called Vickrey-payment), then:

- The mechanism is Individual Rational (IR)
- Non-contributing agents have zero utility
- Mechanism calculates  $V$  and  $V^{-i}$ ,  $i \in N$ :  
 $n + 1$  optimization problems

For some problems: calculation of **two** optimization problems is enough.

# Accelerated computation: LP

- Marginal product:  $V - V^{-i}$ , the ‘price’ of an agent
- Known result in Linear Programming: dual variables correspond to prices
- Idea: find an appropriate LP-formulation

# Accelerated computation: Example

## Assignment problem (Leonard (1983))

- A set of  $n$  agents ( $N$ ) have to be assigned to  $n$  positions ( $M$ )
- Mechanism:
  - Output: an optimal assignment of agents to positions
  - Payments: Vickrey-payments



# Accelerated computation: Example

## LP of Assignment problem

$$\max_x \sum_{i \in N} \sum_{j \in M} v_{ij} \cdot x_{ij}$$

Subject to:

(AP)

$$\sum_{i \in N} x_{ij} \leq 1 \quad \forall j \in M$$

$$\sum_{j \in M} x_{ij} \leq 1 \quad \forall i \in N \quad (1)$$

$$x_{ij} \geq 0 \quad \forall i \in N, \forall j \in M$$

# Accelerated computation: Example

- Remark: integral optimal solution.
- Dual variables of constraints (1) correspond to marginal product
- Problem: multiple optimal solutions possible  
Solution: Leonard (1983) showed which solution



# Accelerated computation: Example

## Dual program of Assignment problem

$$\min_{\pi} \sum_{i \in N} \pi_i^1 + \sum_{j \in M} \pi_j^2$$

Subject to:

(D-AP)

$$\pi_i^1 + \pi_j^2 \geq v_{ij} \quad \forall i \in N, \forall j \in M$$

$$\pi_i^1, \pi_j^2 \geq 0 \quad \forall i \in N, \forall j \in M$$

- $\pi_i^1$  corresponds to marginal product of agent  $i$

# Accelerated computation: Example

Find the MP-yielding solution:

$$\min_{\pi} \sum_{i \in N} \pi_i^1$$

Subject to:

(VCG-D-AP)

$$\sum_{i \in N} \pi_i^1 + \sum_{j \in M} \pi_j^2 = V$$

$$\pi_i^1 + \pi_j^2 \geq v_{ij} \quad \forall i \in N, \forall j \in M$$

$$\pi_i^1, \pi_j^2 \geq 0 \quad \forall i \in N, \forall j \in M$$

# Examples: Combinatorial Auction

- Combinatorial Vickrey Auction
- Bidders bid on packages of items
- Bikhchandani et al. (2002) made an assignment based LP-formulation
- Use the same approach as Leonard (1983) for Assignment problem
- Approach is only valid if *Agents are substitutes*

# Agents are substitutes

Let  $V(K)$  be the maximal welfare that can be achieved if the set of agents  $K \subseteq N$  participates.

$$V(N) - V(K) \geq \sum_{j \in N \setminus K} [V(N) - V(N \setminus j)] \quad \forall K \subseteq N.$$

Meaning: the marginal contribution of a group of agents is more than the sum of the marginal contributions of the individual agents.

# Examples: Minimum spanning tree

- Minimum spanning tree in Graph  $G = (W, E)$
- Agents each own one or more edges
- Mechanism
  - Agents report the cost of an edge
  - The cheapest spanning tree is chosen
  - Agents pay the Vickrey-payments
- Bikhchandani et al. (2002) made an LP-formulation
- If agents are substitutes: Marginal Products in dual solution
- Theorem: Agents are substitutes if no agent owns a *cut*

# Examples: Shortest path problem

- Shortest path in Graph  $G = (W, E)$
- For directed graphs:
  - Bikhchandani et al. (2002) have the Linear programming approach
  - Valid if substitutes property holds
  - Proven for problems that are equivalent with transportation problem

# Examples: Shortest path problem

- For undirected graphs: Hershberger and Suri (2001,2002) use approach that makes use of the graph structure:
  - No substitutes property needed
  - Calculates all payment with the same time complexity as the optimization problem
  - Mechanism needs the same time as two optimization problems

# Approximate Output Algorithms



What if output determination is infeasible?

- use approximation or heuristic

**Example:** combinatorial auction

- Rothkopf et al. (1998): *NP*-hard problem
- Sandholm (2002): inapproximable within reasonable bound



# VCG-based Mechanism

**Definition:** Let  $w = (w^1, \dots, w^n)$  denote a vector of declared types. A direct revelation mechanism  $m = (o, p)$  is called a VCG mechanism based on  $o$  if:

- output function  $o(w)$  maps type declarations into allowable outputs
- payments:  $p^i(w) = \sum_{j \neq i} w^j(o) + h^i(w^{-i})$ , where  $h^i(w^{-i})$  is arbitrary function

**Problem:** VCG-based mechanisms not necessarily truthful

# Example Lehmann et al. (1999)

- Lehmann, O'Callaghan, Shoham (1999)
- combinatorial auction
- set of items  $S$ ,  $n$  bidders
- no externalities:  $v^i(s), \forall s \subseteq S$
- $T^i = \mathbb{R}_+^{2^{|S|}}$
- assume single-minded bidders
- **Definition:** Bidder  $i$  is called single-minded if and only if there exists a set  $s \subseteq S$  and a value  $v \in \mathbb{R}_+$  such that his valuation for  $\tilde{s}$  is given by  $v$  if  $s \subseteq \tilde{s}$  and by 0 otherwise.
- type declaration  $w^i = (s^i, a^i)$

# Greedy Allocation



## Greedy allocation:

- *Phase 1*: (runs in time of order  $n \log n$ )
  - sort type declarations in decreasing order by a norm criterion
  - result: list  $L$
- *Phase 2*: (runs in time linear in  $n$ )
  - allocation generated by greedy algorithm
  - first bid in  $L$  granted
  - following bids in  $L$  examined according to ordering
  - bid granted if not conflicting with previously granted bids
  - otherwise not granted



# Norm Criterion

- should satisfy bid-monotonicity
- **Definition:** The norm of a bid increases if changing  $s$  to  $\tilde{s}$  with  $\tilde{s} \subset s$  or if changing  $a$  to  $\tilde{a}$  with  $\tilde{a} > a$ .
- use average amount per good  $\frac{a}{|s|}$
- $\frac{a}{|s|^{\frac{1}{2}}}$  yields approximation within a factor of  $\sqrt{|S|}$

# Use VCG Payments?

- **Problem:** Greedy allocation and VCG payments do not form a truthful mechanism for single-minded agents.

- **Solution:** non-VCG payment scheme

- let  $w_j = (s_j, a_j)$  be  $j$ th bid in  $L$

- define:  $r(j) = \underbrace{\min}_{A} \{i \mid \underbrace{j < i}_{B}, \underbrace{s_j \cap s_i \neq \emptyset}_{C},$

$\underbrace{\forall l, l < i, l \neq j, l \text{ granted} \Rightarrow s_l \cap s_i = \emptyset}_{D}\}$

- reads:  $r(j)$  is the first bid (A) following  $j$  (B) that has been denied (C) but would have been granted if  $j$  were not there (D).

# Greedy Payment Scheme

**Greedy Payment Scheme:** The payment associated with the  $j$ th bid in  $L$  is calculated as follows:

- If  $w_j$  is not granted or if there is no bid  $r(j)$  then the corresponding agent pays 0.
- If  $w_j$  is granted and there exists a  $r(j)$  then the corresponding agent pays  $|s_j| \frac{a_{r(j)}}{|s_{r(j)}|}$ .

**Theorem:** The mechanism composed of the greedy allocation and payment schemes is truthful for single-minded agents.

# Nisan and Ronen (2000)

- combinatorial auction
- no externalities
- free disposal
- **Definition:** For each agent  $i$  we have that  $v^i(\emptyset) = 0$ .  
Furthermore, take  $s, \tilde{s} \subseteq S$ . If  $s \subseteq \tilde{s}$  then  $v^i(s) \leq v^i(\tilde{s})$ .

# Reasonable Mechanism

- **Definition:** A mechanism for combinatorial auctions is called reasonable if whenever there exists an item  $l \in S$  and an agent  $i \in N$  s.t.
  - For all  $s \subset S$ , if  $l \notin s$  then  $v^i(s \cup \{l\}) > v^i(s)$ .
  - For all agents  $j \neq i$ ,  $v^j(s \cup \{l\}) = v^j(s)$ .then the item  $l$  is allocated to agent  $i$ .
- **Means:** If item desired by only one agent then this agents gets it.
- **Result:** Any reasonable, non-optimal VCG-based mechanism for considered combinatorial auctions is not truthful.

# CMAP



- negative result for cost minimization allocation problems (CMAP)
- example: minimum spanning tree
- degenerate algorithm: solution arbitrarily far from the optimal one
- **Result:** Any non-optimal, non-degenerate VCG-based mechanism for considered CMAPs is not truthful.



# Computationally Limited Agents

- so far: agents can perfectly compute best response functions
- **Strategic knowledge:** describes how agent would like to react in any given situation he can think of
- partial function  $b^i : A^{-i} \mapsto A^i$
- exists at start of game (not gathered during game)
- **Feasible Best Response:**
  - if  $a^{-i}$  not in domain: any  $a^i$
  - if  $a^{-i}$  in domain: best  $a^i$  agent is aware of
- **Feasibly Dominant Action:** feasible best response against all  $a^{-i}$

# Appeals

- **Appeal Function:** An appeal is a function  $l : T \mapsto T$ .
- let  $w = (w^1, \dots, w^n)$  be a type
- agent thinks  $\tilde{w} = (\tilde{w}^1, \dots, \tilde{w}^n)$  yields better outcome
- send appeal  $l(w) = \tilde{w}$
- using this construct a new mechanism out of any VCG-based mechanism

# Second-chance Mechanism

**Definition:** Given an output algorithm  $o(w)$ , the second-chance mechanism looks as follows:

- Each agent reports a type declaration  $w^i$  and an appeal function  $l^i$ , i.e.  $a^i = (w^i, l^i)$ .
- The mechanism computes  $o(w), o(l^1(w)), \dots, o(l^n(w))$  and chooses among these outputs the one that yields the best value for the objective function, i.e. highest social welfare. Denote the chosen output by  $\hat{o}$ .
- The payments are given by  $p^i = \sum_{j \neq i} w^j(\hat{o}) + h^i(w^{-i})$ , where  $h^i(w^{-i})$  is an arbitrary function of  $w^{-i}$ .

# Second-chance Mechanism

- action  $a^i = (w^i, l^i)$  truthful if  $w^i = v^i$
- mechanism feasibly truthful if truth-telling is feasible dominant for all agents
- **Result:** Take a second-chance mechanism with output algorithm  $o$ . For all types  $v \in T$ , if all agents report truthfully then the output chosen by the mechanism is at least as good as  $o(v)$ .
- **Result:** If the output algorithm is computationally feasible and agents have declaration based / appeal independent /  $d$ -obtainable knowledge then the second-chance mechanism is feasibly truthful.

# Declaration Based Knowledge

**Definition:** Knowledge  $b^i$  is called declaration based if it is of the form  $b^i : T^{-i} \mapsto T^i$ .

- agent explores only output algorithm
- agent thinks about declaring  $w^i$
- question for possible types  $w^{-i}$ : Which own report yields best outcome?
- Let's say he thinks: if others have types  $w^{-i}$  then report  $\tilde{w}^i$  better than  $w^i$
- appeal  $l^i(w) = (\tilde{w}^i, w^{-i})$

# Appeal Independent Knowledge

**Definition:** Knowledge  $b^i$  is called appeal independent if it is of the form  $b^i : T^{-i} \mapsto A^i$ .

- agent explores only output algorithm
- question for possible types  $w^{-i}$ : Which report vector yields best outcome?
- Let's say he thinks: if others have types  $w^{-i}$  then report  $\tilde{w} = (\tilde{w}^1, \dots, \tilde{w}^n)$  better than  $w = (w^i, w^{-i})$
- appeal  $l^i(w^i, w^{-i}) = \tilde{w}$

# $d$ -obtainable Knowledge

**Definition:** An agent's knowledge  $b^i$  is called  $d$ -obtainable if:

- $b^i$  is of degree  $d$ .
- Every appeal function that appears, in the domain or range of  $b^i$ , is of degree  $d$ .
- There are at most  $n^d$  appeal functions that appear in the domain or in the range of  $b^i$ . Moreover there exists a representative family  $L^i$  of no more than  $n^d$   $(n - 1)$ -tuples of appeals s.t. for every tuple  $\phi^{-i}$  that appears in the domain of  $b^i$  there exists a  $\psi^{-i} \in L^i$  s.t.  $\forall w^{-i}, b^i((w^{-i}, \phi^{-i})) = b^i((w^{-i}, \psi^{-i}))$ .

# Overview



- Utilitarian problems and the VCG mechanism
  - Accelerated computation of VCG payments
  - Using approximation algorithms
- Non-utilitarian problems
  - Task scheduling
  - Deterministic and randomized algorithms
  - Verification mechanisms
- Distributed algorithmic mechanism design
  - Examples



# Example: Task scheduling

- $k$  tasks,  $n$  agents to do the tasks
- solution: partition  $x$  of tasks over the agents
- $t^i = (t_1^i, \dots, t_k^i)$ :
  - $t_1^i$  time needed by  $i$  to do task  $j$
  - $v^i(t^i, x) = - \sum_{j \in x^i} t_j^i$
- Non-utilitarian objective
  - Minimize the make span
  - $g(x, t) = \max_i \sum_{j \in x^i} t_j^i$

# MinWork



## MinWork mechanism:

- Assign each task to its fastest agent

- Payment: 
$$p^i(t) = \sum_{j \in x^i(t)} \min_{i' \neq i} t_j^{i'}$$

- **Theorem** (Nisan and Ronen (2001))

MinWork is a strongly truthful  $n$ -approximation of the task scheduling problem

- MinWork is polynomial time computable



# Lower bound




**Theorem** (Nisan and Ronen (2001))

There does not exist a mechanism that implements a  $c$ -approximation for the task scheduling problem for any  $c < 2$ .

*Proof:* Construction of counterexample.

**Conjecture** (Nisan and Ronen (2001))

There does not exist a mechanism that implements a  $c$ -approximation for the task scheduling problem for any  $c < n$ .

 (Proven for two special cases.)

# Randomization



## Randomized mechanism:

A randomized mechanism is a probability distribution over a set of mechanisms that have the same sets of strategies and possible outputs.

- Objective:  $E_{m \in M} g(o_m(a), t)$
- $M$  is a set of mechanisms
- $o_m()$  is the output function of mechanism  $m \in M$



# Example: Randomization

## Randomly biased MinWork mechanism:

- Two agents:  $n = 2$
- Randomization: tasks are randomly assigned to agents (50/50)
- Bias:
  - suppose task  $k$  is assigned to agent  $i$
  - Reassign if agent  $j$  is *much* faster
$$t_k^i > \beta \cdot t_k^j$$
  - $\beta = \frac{4}{3}$

# Example: Randomization

**Theorem 4.16** (Nisan and Ronen (2001))

The randomly biased Min Work Mechanism is a

- (polynomial time computable)
- strongly truthful implementation of a
- $\frac{7}{4}$ -approximation

for task scheduling with two agents.

*Proof:* Construction of a worst case example.

# Verification mechanisms



- Verification of agents' reports
- Task scheduling:
  - agents **report** execution times
  - agents actually **do** the jobs
- Idea: payments based on used time
- Strategy: report + execution (agents can slow down)



# Example: verification



## Compensation-and-bonus-mechanism

- Compensation: for time actually **used**
- Bonus: based on own execution and others' reports
- Optimal allocation rule

### **Theorem** (Nisan and Ronen (2001))

The Compensation-and-Bonus mechanism is a strongly truthful implementation of the task scheduling problem.

- Problem: **not** polynomial time computable



# Example: polynomial verification

## Rounding mechanism

- Subclass of task scheduling:
  - Fixed number of agents
  - Bounded type space
- Output: use optimal algorithm for rounded problem
- Payments: rounded version of ‘Compensation and bonus’

## Theorem (Nisan and Ronen (2001))

For every fixed  $\epsilon > 0$  the rounding mechanism is a **polynomial time** mechanism with verification that **truthfully** implements a  $1 + \epsilon$  approximation.

# Distributed Algorithmic MD

- Distributed computation
- $\Rightarrow$  need for communication between agents
- $\Rightarrow$  communication complexity
  - absolute
  - relative
- Feigenbaum et al. (2001), Feigenbaum and Shenker (2002), Nisan (1999)

# Example: Independent set

- $n$  linearly linked processors (agents)
- active agents **exclusively** need both left and right link
- valuation for being active:  $v^i$
- Nisan (1999): two-phase algorithm
  - only communication between directly linked agents
  - gives optimal solution
  - truthful: if only consistent actions
  - otherwise: truth-telling is Nash-equilibrium

# Example: Multicast cost sharing

- Network with some source node  $s$
- Agents are located on nodes
- Valuation for being connected with  $s$
- edge costs  $c(e)$
- Solution: find an optimal tree
- Feigenbaum et al. (2001):
  - Marginal-cost mechanism: VCG, two messages per link
  - Shapley-value mechanism: BB,  $\Theta(nm)$  messages per link

# Example: Interdomain routing

- Extension of the shortest path problem
- Routing of packages between Autonomous systems
- Feigenbaum et al. (2002): distributed mechanism based on BGP-protocol
  - VCG
  - roughly the same communication complexity as BGP-protocol