Game-theoretic Mechanisms

Elias Koutsoupias

University of Athens
http://www.di.uoa.gr/~elias

Reykjavik 2008/07/04-06

Outline

Outline

Why Game Theory and CS?

Applications

- Internet routing (interactions between ISPs)
- Search ads (e.g., adwords, adsense)
- Online auctions (e.g. Ebay)
- P2P (e.g. free-riders)
- ...

Algorithmic Game Theory

Main research directions

- Computational issues of games (e.g., finding any Nash equilibrium is PPAD-complete)
- Price of Anarchy (Inefficiency of systems with selfish entities)
- Algorithms + Incentives = Mechanisms

Algorithmic Game Theory

Main research directions

- Computational issues of games (e.g., finding any Nash equilibrium is PPAD-complete)
- Price of Anarchy (Inefficiency of systems with selfish entities)
- Algorithms + Incentives = Mechanisms

Notions of equilibria

Nash equilibrium

No player has reasons to deviate.

- Advantages: Natural. It always exists.
- Disadvantages: More than one. Computationally hard?

Dominant strategies

Each player has an optimal strategy, no matter what.

- Advantages: Natural. Great for implementation.
- Disadvantages: Rarely exists.

Outline

Mechanism design

Mechanisms as Algorithms

- Given an objective, design **a game** whose equilibrium is the objective.
- Here we consider dominant equilibria (i.e., a player has an optimal strategy, no matter what the other players do).

Typical Example: Single-item Auction

Problem

- *We want to sell an object to n players (buyers).*
- *Each player has a value v_i for the object, which is known only to him/her .*
- *Objective: Give the item to the player with the highest value.*

Features

- Incomplete information: only the players know their values
- Money may be used as an incentive. But: money is not part of the objective.
- Direct revelation: The players declare all their values at the beginning.

Example: Single-item Auction (cont.)

The VCG mechanism

- Each player declares a value $\hat{v}_i$, not necessarily equal to the true value v_i.
- The mechanism allocates the object to the player with the highest bid, $\max_i \hat{v}_i$. **This is the objective when the players are truthful.**
- The player pays only the second highest bid.

Proposition

The VCG mechanism is truthful.

Why is it truthful?

- The payment depends only on the other players
- The allocation is monotone: increasing the declared value makes more likely to get the item

The Social Choice Framework

The social choice setting

- There are n players and m outcomes. Let v_{ij} be the gain of player i when the outcome of the game is j.

$$\begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \end{bmatrix}$$

- The domain D of the problem is a set of $n \times m$ matrices.
- The objective of the mechanism designer is to select the outcome (i.e., column) which optimizes his/her objective.
- The objective of each player is to maximize his/her gain.
- Only the players know the values v_{ij}.

The Social Choice Framework (cont.)

Problem (The single-item auction)

There are n players and $m = n$ outcomes. The i-th outcome is for player i to get the item.
The domain of the problem is all $n \times n$ matrices of the form

$$\begin{bmatrix} v_1 & 0 & 0 \\ 0 & v_2 & 0 \\ 0 & 0 & v_3 \end{bmatrix}$$

Each row corresponds to a player, and each column to an outcome.

Combinatorial Auction

Problem (Combinatorial auction)

- *There are n players (bidders) and m objects (items)*
- *Each player i has a value $u_{i,S}$ for each subset (bundle) S of the objects. These are private values.*
- *Objective: Allocate the objects to the players to maximize the sum of the values of their bundles.*

Example (3 players, 2 items)

$$\begin{bmatrix} u_{1,12} & u_{1,1} & u_{1,1} & u_{1,2} & u_{1,2} & 0 & 0 & 0 & 0 \\ 0 & u_{2,2} & 0 & u_{2,1} & 0 & u_{2,12} & u_{2,1} & u_{2,2} & 0 \\ 0 & 0 & u_{3,2} & 0 & u_{3,1} & 0 & u_{3,2} & u_{3,1} & u_{3,12} \end{bmatrix}$$

Scheduling unrelated machines

Problem (Scheduling)

- *There are n players (machines) and m objects (tasks)*
- *Each player i has a (private) value t_{ij} for each task j*
- *Objective: Allocate the tasks to the players to minimize the maximum value among the players (i.e., the makespan)*

Example (2 players, 2 tasks)

$$\begin{bmatrix} t_{11}+t_{12} & t_{11} & t_{12} & 0 \\ 0 & t_{22} & t_{21} & t_{21}+t_{22} \end{bmatrix}$$

Cost Sharing

Problem (Budget-balanced cost sharing)

- *There are n players. For each subset S of players there is cost c_S.*
- *Each player i has a (private) value t_i*
- *Objective: Select a maximum-cardinality set of players whose values cover the cost.*

Example (2 players)

$$\begin{bmatrix} 0 & t_1 & 0 & t_1 \\ 0 & 0 & t_2 & t_2 \end{bmatrix}$$

Direct Revelation Mechanisms

The protocol of the mechanism

Declare Each player i declares his/her values $\hat{v}_{ij}$.

Allocate An allocation algorithm A computes the outcome $j^* = A(\hat{v})$.

Pay A payment algorithm p computes for each player i a payment $p_i(\hat{v}, j^*)$.

The objectives

Player Player i gains $v_{ij^*} - p_i(\hat{v}, j^*)$.

Social The objective of the mechanism is to select the outcome j^* which optimizes some global objective $f(v)$. (For example to select the column with maximum total value).

Truthful mechanisms

Definition (Truthful mechanisms)

A mechanism is truthful when revealing the true values ($\hat{v}_{ij} = v_{ij}$) is a dominant strategy of every player.

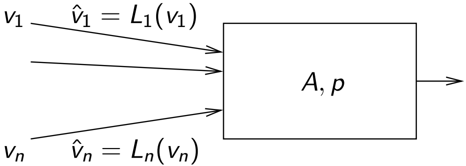
Theorem (The revelation principle)

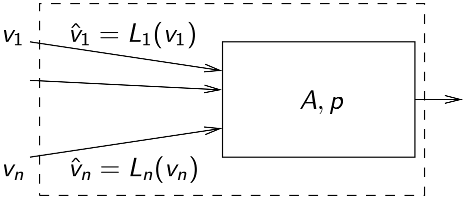
For every mechanism there is an equivalent truthful mechanism (with the same payments and outcome) .

Why?

Given a non-truthful mechanism, we can design a new truthful mechanism which first simulates the lying strategies of the players and then applies the original mechanism. The players would tell the truth to this mechanism.

The revelation principle





(A Parenthesis) Why payments?

Theorem (Gibbard-Satterthwaite)

Only dictatorships are truthful!

In other words, no non-trivial social choice functions have truthful mechanisms without payments.
On the other hand, there are some very interesting truthful mechanisms without payments: For example the stable matching algorithm.

Outline

Truthful mechanisms for single-item auction

First Price

- The mechanism in which the highest bidder gets the item and pays his declared price is **not truthful**.
- Counterexample: $v_1 = 2$, $v_2 = 1$. Player 1 gains by bidding $\hat{v}_1 = 1 + \epsilon$.

Second Price

- The mechanism in which the highest bidder gets the item and pays the second highest price is **truthful**.

Central question

Which mechanisms are truthful?

Which mechanisms are truthful?

Focus on allocations

- The objective (social choice function) does not involve the payments.
- Which allocation algorithms admit a payment policy that makes the mechanism truthful?

Example (Single-item auction)

- The algorithm which allocates the object to the **highest value** is truthful. (The second price payment policy makes it **truthful**).
- The algorithm which allocates the object to the **second highest value** is **not truthful**. (There is no payment policy to make it truthful). Why?

The VCG and the affine maximizer

Definition (VCG)

The Vickrey-Clarke-Groves (VCG) mechanism selects the outcome which maximizes the **sum of the values** of the players.

Definition (Affine maximizer)

In an affine maximizer (or generalized VCG) there are constants λ_i (one for each player) and γ_j (one for each outcome) and the mechanism selects the outcome j which maximizes $\sum_i \lambda_i v_{ij} + \gamma_j$.

Example (Affine maximizer for 2 players, 3 outcomes)

$$\begin{array}{cccccc}
v_{11} & v_{12} & v_{13} & \leftarrow & \lambda_1 \\
v_{21} & v_{22} & v_{23} & \leftarrow & \lambda_2 \\
\uparrow & \uparrow & \uparrow & & \\
\gamma_1 & \gamma_2 & \gamma_3 & &
\end{array}$$

The VCG Mechanism

Theorem

The affine maximizers mechanisms (generalized VCG) are truthful.

- The payment of each player i is equal to the (weighted) sum of the remaining players plus an arbitrary value that depends on the values of the other players:

$$\lambda_i p_i(v, j) = -\sum_{i' \neq i} \lambda_{i'} v_{i'j} + h_i(v_{-i})$$

- The objective (value + payment) of each player i becomes (almost) identical to the global objective!
- We can think of it, as giving a discount to a player equal to the increase of the global objective because of his/her participation (by carefully selecting the function h).

The VCG Mechanism for the combinatorial auction

Is VCG good?

- For the combinatorial auction problem, where the global objective is to maximize the total value, the VCG achieves the global objective.
- There is however a problem: Computing the optimal solution may be computationally hard.
  - If the input is the whole $n \times k^n$ array, then the problem is computationally trivial (linear-time).
  - If the input is given implicitly, then the problem can be NP-hard.

The VCG Mechanism for the scheduling problem

VCG does not match the social optimum

- The VCG mechanism is not appropriate for the scheduling problem. It maximizes the sum, while the objective is the makespan!

Comparison of combinatorial auctions and scheduling

- The domain of scheduling is a **restriction** of the domain of combinatorial auction in which the valuations of bundles are additive.
- Auction is a maximization problem, scheduling is a minimization problem. (Not a significant difference.)
- They differ in the objective. One aims at the sum the other at the max. In this respect, scheduling is more difficult.

Outline

Characterization of truthful mechanisms

Let $x_j = x_j(v)$ be a 0-1 value that indicates the selected outcome.

$$x_j = \begin{cases} 1 & \text{if the allocation algorithm selects outcome } j \\ 0 & \text{otherwise} \end{cases}$$

Definition (Monotonicity)

An allocation algorithm is called monotone if for every two inputs v and v' that **differ only on the i-th player**, the allocations x and x' satisfy

$$\sum_j (x_j - x'_j)(v_{ij} - v'_{ij}) \geq 0$$

This is sometimes called weak monotonicity in the literature.

Truthful ⇒ Monotone

Proof of necessity

- Fix a truthful mechanism. The payment of player i should be independent of his declaration: $p_i = p_i(x, v_{-i})$.
- When player i has value v_i, he has no reason to declare v_i':

$$\sum_j x_j v_{ij} - p_i(x, v_{-i}) \geq \sum_j x_j' v_{ij} - p_i(x', v_{-i})$$

When he has value v_i', he has no reason to declare v_i:

$$\sum_j x_j' v_{ij}' - p_i(x', v_{-i}) \geq \sum_j x_j v_{ij}' - p_i(x, v_{-i})$$

If we add them, we get the Monotonicity condition.

Truthful ⇔ Monotone

Theorem (Saks-Yu, 2005)

Monotonicity is necessary and sufficient condition for truthfulness for convex domains.

- The proof of sufficiency is non-trivial.
- The characterization applies to almost all interesting problems with continuous domains.
- It does not apply to discrete domains. For example, when there are two possible values for each item, low and high.
- This characterization is complete but not as useful as a global characterization.

Roberts' Theorem

Theorem (Roberts, 1979)

For the unrestricted domain with at least 3 outcomes, the only truthful mechanisms are the affine maximizers.

Desired characterization

- This characterization is much more useful than the monotonicity property.
- Can we get similar characterizations for other domains/problems such as combinatorial auctions and scheduling?
  - The Monotonicity property is enough for the single-parameter domains
  - For multi-parameter domains, the question is open

Characterization - Auctions and scheduling

- For combinatorial auctions, under certain assumptions, truthful mechanisms are only the affine maximizers (Lavi, Mu'alem, Nisan 2003).
- For the 2-player scheduling problem, truthful mechanisms with bounded approximation ratio are only the affine minimizers (Dobzinski, Sundararajan)
- For the 2 player scheduling problem, truthful mechanisms are only the affine minimizers and task-independent algorithms (Christodoulou, K, Vidali)

Conjecture

For any number of players for the scheduling problem (and therefore for the combinatorial auction domain), truthful mechanisms are only the affine minimizers and threshold algorithms.

Characterization - Discrete domains

- The Monotonicity property is not sufficient to capture truthfulness for discrete domains.
- A necessary and sufficient condition is Cycle Monotonicity: Fix the values v_{-i} of all players except i and consider values $v_i^1, \ldots, v_i^k, v_i^{k+1} = v_i^1$ for player i. Let $x^1, \ldots, x^k, x^{k+1}$ be the outcome of the mechanism for these values. Then we must have:

$$\sum_{r=1}^{k} \sum_{j} v_{ij}^r (x_j^r - x_j^{r+1}) \geq 0.$$

Outline

Outline

VCG Combinatorial auctions

- The VCG mechanism is truthful and optimal for combinatorial auctions
- When the input is given implicitly, the VCG is an exponential-time algorithm
- Is there a truthful mechanism which runs in polynomial time for implicit inputs and is optimal or has good approximation ratio?
- A good example is a greedy algorithm by Lehmann, Callaghan,and Shoham.

Single-minded auctions

- Player i is single-minded when he cares about only one bundle S_i for which it has value v_i.
- The input to this combinatorial auction problem is $(S_1, v_1), \ldots, (S_n, v_n)$ and we seek an algorithm that partitions the items optimally.
- The offline problem is NP-hard. It is even NP-hard to approximate the optimal value within a factor less than $\sqrt{m}$, where m is the number of items.

Single-minded auctions

- There is a mechanism that runs in polynomial time and guarantees approximation ratio $\sqrt{m}$:
- Reorder first the players so that

$$\frac{v_1}{\sqrt{|S_1|}} \geq \cdots \geq \frac{v_n}{\sqrt{|S_n|}}$$

- Process the players in order: Give a player its bundle if all its items are unallocated, otherwise give nothing.

Theorem

This algorithm runs in polynomial time, is truthful, and achieves an approximation ratio $\sqrt{m}$.

Outline

Related machines scheduling

- In this scheduling problem, machines differ only in their speeds (which are their private value).
- This is a typical **single-parameter domain**.
- The problem is NP-hard, but there is a PTAS (it can be approximated to within any ϵ in polynomial time).
- There is an optimal algorithm which is monotone (and therefore truthful), but it takes exponential time.
- The interesting question is whether there exists a PTAS or a polynomial-time approximation truthful mechanism.

Related machines scheduling

Results

- Archer and Tardos gave a variant of the (exponential-time) optimal algorithm which is truthful
- They also gave a polynomial-time randomized 3-approximation mechanism, which was later improved by Archer to 2-approximation.

Related machines scheduling

Results

- Andelman, Azar, and Sorani gave a 5-approximation deterministic truthful mechanism.
  The main idea was to find an optimal monotone fractional solution and to round it (with randomized rounding within a factor of 2 or deterministic rounding within a factor of 5)
- Kovács improved it to 3-approximation and later to 2.8.
  The main idea is to round every value to the next power of 2 and to show that the greedy (LPT) algorithm for these values is monotone (and therefore truthful).
- A very recent result (by Dhangwatnotai, Dobzinski, Dughmi and Roughgarden) gives a randomized PTAS for the problem.
- Is there a deterministic PTAS?

Outline

Upper bound for unrelated machines scheduling

- The unrelated machines scheduling is an NP-hard problem. It is even NP-hard to approximate it to within a factor of 3/2.
- It can be approximated in polynomial time within a factor of 2.
- How well can we approximate it with mechanisms (even when with exponential-time ones)?

Theorem (Nisan, Ronen)

The VCG mechanism has approximation ratio n.

Proof: The VCG returns the optimal solution when the objective is the total welfare. On the other hand, the objective for the scheduling is the maximum welfare. The sum is trivially within a factor of n from the maximum.

Outline

Discrete domain with only 2 values

- For the special case of the scheduling problem on discrete domains, truthfulness is characterized by cycle monotonicity.
- Lavi and Swamy gave a deterministic 2-approximation mechanism for the case of 2 values (L and H).

Discrete domain with only 2 values (cont.)

- The main idea is to take any approximation algorithm and transform it to a fractional algorithm which satisfies the cycle monotonicity property, by reallocated the tasks.
- To guarantee that the transformation works they use a trick:
  If for a task j a machine has high value, then the machine gets at most $1/n$ of the task. If it has low value, then it gets at least $1/n$ of the task.
- This can be accomplished by reallocated the tasks as follows: We take every task allocated to a machine with high value and spread it to all machines. We take every task allocated to a low machine and spread (part of) it to low machines.

Outline

The fractional scheduling

Fractional allocations

- With fractional allocations each task can be split across the machines.
- The classical version of the problem is solvable in polynomial time (by linear programming).
- fractional approximation ratio $\leq$ randomized approximation ratio

Fractional Version: Upper Bound

The SQUARE Algorithm

The mechanism SQUARE is a task independent algorithm which allocates to every player i a fraction inversely proportional to t_{ij}^2 of task j.

Theorem (Christodoulou, K, Kovàcs)

The mechanism SQUARE is truthful with approximation ratio $\frac{n+1}{2}$.

Fractional Version: Upper Bound (cont.)

Ingredients of the proof

- Assume wlog that the optimum allocation is integral.
- Let $S_1, \ldots, S_n$ be an optimal allocation.
- We consider the execution time of SQUARE for machine i:

$$cost_i = \sum_j x_{ij} t_{ij} = \sum_r \sum_{j \in S_r} x_{ij} t_{ij}$$

- We will show that

$$\sum_{j \in S_r} x_{ij} t_{ij} \leq opt_r \qquad \text{for } r = i$$

$$\sum_{j \in S_r} x_{ij} t_{ij} \leq \frac{1}{2} opt_r \qquad \text{for } r \neq i$$

Fractional Version: Upper Bound (cont.)

- For $r = i$, we have $\sum_{j \in S_r} x_{ij} t_{ij} \leq \sum_{j \in S_r} t_{ij} = opt_r$.
- For $r \neq i$, we have

$$\begin{aligned}
\sum_{j \in S_r} x_{ij} t_{ij} &= \sum_{j \in S_r} \frac{\frac{1}{t_{ij}^2}}{\sum_k \frac{1}{t_{kj}^2}} t_{ij} \\
&\leq \sum_{j \in S_r} \frac{\frac{1}{t_{ij}^2}}{\frac{1}{t_{rj}^2} + \frac{1}{t_{ij}^2}} t_{ij} \\
&= \sum_{j \in S_r} \frac{t_{ij} t_{rj}}{t_{ij}^2 + t_{rj}^2} t_{rj} \\
&\leq \sum_{j \in S_r} \frac{1}{2} t_{rj} \\
&= \frac{1}{2} opt_r
\end{aligned}$$

Outline

Scheduling unrelated machines

Definition

- There are n players (machines) and m objects (tasks)
- Each player i has a (private) value t_{ij} for each task j
- Objective: Allocate the tasks to the players to minimize the maximum value among the players (i.e., the makespan)

The setting

Input

$$t = \begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1m} \\ t_{21} & t_{22} & \cdots & t_{2m} \\ \cdots \\ t_{n1} & t_{n2} & \cdots & t_{nm} \end{pmatrix} \longrightarrow$$

Output

$$x = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \cdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix}$$

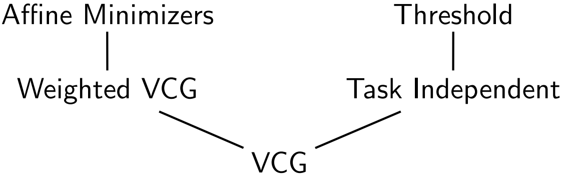
Input – Output

- n players/machines (rows).
- m tasks (columns).
- The input consists of nonnegative values t_{ij}.
- The output is an allocation:

$$x_{ij} = \begin{cases} 1 & \text{when task } j \text{ is allocated to machine } i \\ 0 & \text{otherwise} \end{cases}$$

Simple algorithms

VCG and its generalizations



VCG: Allocate each task to the machine of minimum value

Weighted VCG: VCG but first speedup some machines

Affine Minimizer: Weigthed VCG with additional constants for each allocation

Task Independent: Allocate every task independently of the others

Threshold: Allocate a task j to machine i independently of the other values of machine i

Lower bound n for these algorithms

Theorem

All affine minimizers and threshold algorithms have approximation ratio at least n.

Threshold algorithms.

$n(n-1)+1$ tasks (all values equal). Some player (say the first) gets at least $n+1$ tasks.

$$t = \begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & \mathbf{1} & \mathbf{1} & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & \mathbf{1} & \mathbf{1} \end{pmatrix}$$

Change the value of the remaining tasks to 0. The allocation of the first player for these tasks remains the same (by the definition of threshold algorithms).

Lower bound n for these algorithms

Theorem

All affine minimizers and threshold algorithms have approximation ratio at least n.

Threshold algorithms.

$n(n-1)+1$ tasks (all values equal). Some player (say the first) gets at least $n+1$ tasks.

$$t = \begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & \mathbf{1} & \mathbf{1} & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & \mathbf{1} & \mathbf{1} \end{pmatrix}$$

Change the value of the remaining tasks to 0. The allocation of the first player for these tasks remains the same (by the definition of threshold algorithms).

$$t' = \begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & \mathbf{0} \\ 1 & 1 & 1 & \mathbf{1} & 1 & \mathbf{1} & 1 \\ 1 & 1 & 1 & 1 & \mathbf{1} & 1 & 1 \end{pmatrix}$$

Truthful ≡ Monotone

Definition (Monotonicity Property)

An allocation algorithm is called monotone if it satisfies the following property: for every two sets of tasks t and t' which differ only on machine i (i.e., on the i-the row) the associated allocations x and x' satisfy

$$(x_i - x_i') \cdot (t_i - t_i') \leq 0,$$

where $\cdot$ denotes the dot product of the vectors, that is, $\sum_{j=1}^{m}(x_{ij} - x_{ij}')(t_{ij} - t_{ij}') \leq 0$.

Theorem (Nisan, Ronen 1998, Saks, Lan Yu 2005)

$$\textit{Truthful} \equiv \textit{Monotone}$$

The Monotonicity Property

First Input

$$\begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1m} \\ \cdots & & & \\ t_{i1} & t_{i2} & \cdots & t_{im} \\ \cdots & & & \\ t_{n1} & t_{n2} & \cdots & t_{nm} \end{pmatrix} \Rightarrow \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ \cdots & & & \\ x_{i1} & x_{i2} & \cdots & x_{im} \\ \cdots & & & \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix}$$

Second Input

$$\begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1m} \\ \cdots & & & \\ t'_{i1} & t'_{i2} & \cdots & t'_{im} \\ \cdots & & & \\ t_{n1} & t_{n2} & \cdots & t_{nm} \end{pmatrix} \Rightarrow \begin{pmatrix} x'_{11} & x'_{12} & \cdots & x'_{1m} \\ \cdots & & & \\ x'_{i1} & x'_{i2} & \cdots & x'_{im} \\ \cdots & & & \\ x'_{n1} & x'_{n2} & \cdots & x'_{nm} \end{pmatrix}$$

Monotonicity

$$\sum_{j=1}^{m} (x_{ij} - x'_{ij})(t_{ij} - t'_{ij}) \leq 0$$

A geometric approach to truthfulness

2 tasks

- Fix all values except of t_{11} and t_{12}. Consider how the space of t_{11} and t_{12} is partitioned by a truthful mechanism.
- R_{ab}: the region for which the allocation of the first machine is $x_{11} = a$ and $x_{12} = b$.
- The Monotonicity Property implies that a mechanism is truthful iff the regions R_{ab} and $R_{a'b'}$ are separated by a line of the form

$$(a - a')t_{11} + (b - b')t_{12} = const.$$

A geometric approach to truthfulness

The geometry of truthful mechanisms

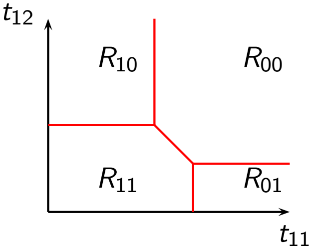


Figure: The two possible ways to partition the positive orthant.

Boundaries for the special cases

- For affine maximizers, the boundaries depend linearly on the values of the other players, and the diagonal part has constant length
- For threshold mechanisms, the diagonal part does not exist but the boundaries can be arbitrary (monotone) functions.

History of the scheduling problem

Computational issues

- It is a well-studied NP-hard problem
- Lenstra, Shmoys, and Tardos showed that its approximation ratio is in $[1.5, 2]$.

Mechanisms for scheduling

- Nisan and Ronen in 1998 initiated the study of its mechanism-design version.
- They gave an **upper bound** (VCG) with approximation ratio n.
- They gave a **lower bound** of 2.
- They conjectured that the right answer is the upper bound.
- They also gave a randomized mechanism with approximation ratio $7/4$ for 2 players.

Recent results

Deterministic

- The lower bound was improved to 2.41 (Christodoulou – K – Vidali) and to 2.61 (K – Vidali).
- For 2 machines the only truthful mechanisms with bounded approximation ratio are task-independent (Dobzinski – Sundararajan).
- For 2 machines, (with some mild asumptions) the only truthful mechanims are either affine minimizers or task-independent (Christodoulou – K – Vidali, submitted).

Recent results

Randomized

- The lower bound was improved to $2 - 1/n$ (Mu'alem – Schapira).
- The upper bound was improved to $7n/8$ (Mu'alem – Schapira).

Fractional

- The lower bound was improved to $2 - 1/n$ (Christodoulou – K – Kovács).
- The upper bound for task-independent mechanisms was pinned to $(n+1)/2$ (Christodoulou – K – Kovács).

Discrete (high and low value)

- Mechanism with approximation ratio 2 (Lavi – Swamy).

Outline

How to use the Monotonicity Property

We manipulate the values of one player in a particular way which guarantees that his allocation remains the same.

Example (Change the values, keep the allocation)

$$t = \begin{pmatrix} \mathbf{1} & 2 & \mathbf{2} \\ 2 & \mathbf{3} & 1 \\ 1 & 2 & 2 \end{pmatrix}$$

How to use the Monotonicity Property

We manipulate the values of one player in a particular way which guarantees that his allocation remains the same.

Example (Change the values, keep the allocation)

$$t = \begin{pmatrix} \mathbf{1} & 2 & \mathbf{2} \\ 2 & \mathbf{3} & 1 \\ 1 & 2 & 2 \end{pmatrix} \quad \rightarrow \quad t' = \begin{pmatrix} \mathbf{1} - \epsilon_1 & 2 + \epsilon_2 & \mathbf{2} - \epsilon_3 \\ 2 & 3 & 1 \\ 1 & \mathbf{2} & 2 \end{pmatrix}$$

How to use the Monotonicity Property

We manipulate the values of one player in a particular way which guarantees that his allocation remains the same.

Example (Change the values, keep the allocation)

$$t = \begin{pmatrix} \mathbf{1} & 2 & \mathbf{2} \\ 2 & \mathbf{3} & 1 \\ 1 & 2 & 2 \end{pmatrix} \quad \rightarrow \quad t' = \begin{pmatrix} \mathbf{1} - \epsilon_{\mathbf{1}} & 2 + \epsilon_2 & \mathbf{2} - \epsilon_{\mathbf{3}} \\ 2 & 3 & 1 \\ 1 & \mathbf{2} & 2 \end{pmatrix}$$

Example (Increase a value, keep the allocation)

$$t = \begin{pmatrix} \mathbf{0} & \cdots \\ \infty & \cdots \\ \infty & \cdots \end{pmatrix}$$

How to use the Monotonicity Property

We manipulate the values of one player in a particular way which guarantees that his allocation remains the same.

Example (Change the values, keep the allocation)

$$t = \begin{pmatrix} \mathbf{1} & 2 & \mathbf{2} \\ 2 & \mathbf{3} & 1 \\ 1 & 2 & 2 \end{pmatrix} \quad \rightarrow \quad t' = \begin{pmatrix} \mathbf{1-\epsilon_1} & 2+\epsilon_2 & \mathbf{2-\epsilon_3} \\ 2 & 3 & 1 \\ 1 & \mathbf{2} & 2 \end{pmatrix}$$

Example (Increase a value, keep the allocation)

$$t = \begin{pmatrix} \mathbf{0} & \cdots \\ \infty & \cdots \\ \infty & \cdots \end{pmatrix} \quad \rightarrow \quad t' = \begin{pmatrix} \mathbf{1} & \cdots \\ \infty & \cdots \\ \infty & \cdots \end{pmatrix}$$

Easy proof of lower bound 2

2 players, 3 tasks

Either the mechanism partitions the tasks to the two machines

$$\begin{pmatrix} \color{red}{\mathbf{1}} & 1 & 1 \\ 1 & \color{red}{\mathbf{1}} & \color{red}{\mathbf{1}} \end{pmatrix}$$

or gives all tasks to the same machine

$$\begin{pmatrix} 1 & 1 & 1 \\ \color{red}{\mathbf{1}} & \color{red}{\mathbf{1}} & \color{red}{\mathbf{1}} \end{pmatrix}$$

Easy proof of lower bound 2

2 players, 3 tasks

Either the mechanism partitions the tasks to the two machines

$$\begin{pmatrix} \mathbf{1} & 1 & 1 \\ 1 & \mathbf{1} & \mathbf{1} \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} \mathbf{0} & 1 & 1 \\ 1 & \mathbf{1} & \mathbf{1} \end{pmatrix}$$

or gives all tasks to the same machine

$$\begin{pmatrix} 1 & 1 & 1 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} 1 & 1 & 1 \\ \mathbf{0} & \mathbf{1} & \mathbf{1} \end{pmatrix}$$

The instances of the 2.61 lower bound

$$\begin{pmatrix} 0 & \cdots & \infty & a & a^2 & \cdots & a^{n-1} \\ \infty & \cdots & \infty & a^2 & a^3 & \cdots & a^n \\ \cdots \\ \infty & \cdots & 0 & a^n & a^{n+1} & \cdots & a^{2n-1} \end{pmatrix}$$

Claim

If the first player does not get all the non-dummy tasks (the a^j tasks), then the approximation ratio is at least $1 + a$.

Therefore the approximation ratio is

$$\min\{1 + a, \frac{a + a^2 + \cdots + a^{n-1}}{a^{n-1}}\}.$$

For $n \to \infty$ and $a = \phi$, the ratio is $2.618\ldots$.

The Proof of the Claim

- We prove the claim by induction. For this we need to strengthen the induction hypothesis. The claim holds for all instances of the form

$$\begin{pmatrix} 0 & \cdots & \infty & a^{i_1} & a^{i_2} & \cdots & a^{i_k} \\ \infty & \cdots & \infty & a^{i_1+1} & a^{i_2+1} & \cdots & a^{i_k+1} \\ \cdots & & & & & & \\ \infty & \cdots & 0 & a^{i_1+n-1} & a^{i_2+n} & \cdots & a^{i_k+n-1} \end{pmatrix}$$

- $k \in \{1, \ldots, n-1\}$ and $i_1 < i_2 < \cdots < i_k$.

The Proof of the Claim (cont.)

Manipulating the values

- Assume that the first player does not get all the non-dummy tasks.
- We first manipulate the values so that
  - the **first player** gets **no** non-zero task and
  - **every other player** gets **at most one** non-zero task.

The Proof of the Claim (cont.)

Example (No task for the first player)

$$\begin{pmatrix} 0 & \cdots & \infty & a^{i_1} & a^{i_2} & \cdots & \mathbf{a^{i_k}} \\ \infty & \cdots & \infty & a^{i_1+1} & a^{i_2+1} & \cdots & a^{i_k+1} \\ \cdots & & & & & & \\ \infty & \cdots & 0 & a^{i_1+n-1} & a^{i_2+n-1} & \cdots & a^{i_k+n-1} \end{pmatrix}$$

If the first player gets some non-zero task, reduce the value to 0. The first player keeps the same tasks (by Monotonicity).

$$\begin{pmatrix} 0 & \cdots & \infty & a^{i_1} & a^{i_2} & \cdots & \mathbf{0} \\ \infty & \cdots & \infty & a^{i_1+1} & a^{i_2+1} & \cdots & a^{i_k+1} \\ \cdots & & & & & & \\ \infty & \cdots & 0 & a^{i_1+n-1} & a^{i_2+n-1} & \cdots & a^{i_k+n-1} \end{pmatrix}$$

The Proof of the Claim (cont.)

Example (Zero or one task for other players)

$$\begin{pmatrix} 0 & \cdots & \infty & a^{i_1} & a^{i_2} & \cdots & a^{i_k} \\ \infty & \cdots & \infty & \mathbf{a^{i_1+1}} & \mathbf{a^{i_2+1}} & \cdots & a^{i_k+1} \\ \cdots \\ \infty & \cdots & 0 & a^{i_1+n-1} & a^{i_2+n-1} & \cdots & a^{i_k+n-1} \end{pmatrix}$$

If some other player gets at least two non-zero tasks, reduce one value to 0. The player still gets at least one non-zero task.

$$\begin{pmatrix} 0 & \cdots & \infty & a^{i_1} & a^{i_2} & \cdots & a^{i_k} \\ \infty & \cdots & \infty & \mathbf{0} & \mathbf{a^{i_2+1}} & \cdots & a^{i_k+1} \\ \cdots \\ \infty & \cdots & 0 & a^{i_1+n-1} & a^{i_2+n-1} & \cdots & a^{i_k+n-1} \end{pmatrix}$$

The Proof of the Claim (cont.)

The result

At the end of the process,

- the first player has no non-zero tasks,
- every other player has at most one non-zero task,
- some other player has exactly one non-zero task.

The Proof of the Claim (cont.)

Estimating

- The optimum value is a^{i_k} (the diagonal right-to-left).
- We find a task with cost at least a^{i_k+1} and we raise its dummy (diagonal) value to a^{i_k}.
- The heart of the proof is that there always exists such a task which will not affect the optimum value.
- The cost of the mechanism is at least $a^{i_k} + a^{i_k+1}$ while the optimum is a^{i_k}. The approximation ratio is at least $1 + a$.

Example

$$\begin{pmatrix} 0 & \infty & \infty & \cdots & a^{i_k-3} & a^{i_k-1} & a^{i_k} \\ \infty & 0 & \infty & \cdots & a^{i_k-2} & \mathbf{a^{i_k}} & a^{i_k+1} \\ \infty & \infty & 0 & \cdots & a^{i_k-1} & a^{i_k+1} & \mathbf{a^{i_k+2}} \\ \cdots & & & & & & \end{pmatrix}$$

The Proof of the Claim (cont.)

Estimating

- The optimum value is a^{i_k} (the diagonal right-to-left).
- We find a task with cost at least a^{i_k+1} and we raise its dummy (diagonal) value to a^{i_k}.
- The heart of the proof is that there always exists such a task which will not affect the optimum value.
- The cost of the mechanism is at least $a^{i_k} + a^{i_k+1}$ while the optimum is a^{i_k}. The approximation ratio is at least $1 + a$.

Example

$$\begin{pmatrix} 0 & \infty & \infty & \cdots & a^{i_k-3} & a^{i_k-1} & a^{i_k} \\ \infty & 0 & \infty & \cdots & a^{i_k-2} & a^{i_k} & a^{i_k+1} \\ \infty & \infty & \mathbf{a^{i_k}} & \cdots & a^{i_k-1} & a^{i_k+1} & \mathbf{a^{i_k+2}} \\ \cdots \end{pmatrix}$$

Outline

1. CS and GT
2. Social Choice Framework
3. VCG and Affine maximizers
4. Characterization of truthful mechanisms
5. Positive results
   - Single-minded auctions
   - Related machines
   - Unrelated machines
     - Discrete domain
     - Fractional algorithms
6. Lower bounds
   - Deterministic lower bound
   - Fractional lower bound
7. Open problems
8. Similar topics
   - Sponsored Search
   - Competitive auctions

The Fractional Version

Fractional allocations

- With fractional allocations each task can be split across the machines.
- The classical version of the problem is solvable in polynomial time (by linear programming).
- fractional approximation ratio $\leq$ randomized approximation ratio

Fractional Version: Lower Bound

A bad input

$$\begin{pmatrix} 0 & \infty & \cdots & \infty & \cdots & \infty & n-1 \\ \infty & 0 & \cdots & \infty & \cdots & \infty & n-1 \\ \cdots & & & & & & \\ \infty & \infty & \cdots & 0 & \cdots & \infty & n-1 \\ \cdots & & & & & & \\ \infty & \infty & \cdots & \infty & \cdots & 0 & n-1 \end{pmatrix}$$

Proving a lower bound of $2 - 1/n$

- Find the player who gets the largest fraction of the last task and raise its diagonal 0 value to 1.

Fractional Version: Lower Bound

A bad input

$$\begin{pmatrix} 0 & \infty & \cdots & \infty & \cdots & \infty & n-1 \\ \infty & 0 & \cdots & \infty & \cdots & \infty & n-1 \\ \cdots \\ \infty & \infty & \cdots & \mathbf{1} & \cdots & \infty & \mathbf{n-1} \\ \cdots \\ \infty & \infty & \cdots & \infty & \cdots & 0 & n-1 \end{pmatrix}$$

Proving a lower bound of $2 - 1/n$

- Find the player who gets the largest fraction of the last task and raise its diagonal 0 value to 1.
- When we change the values, the allocation remains almost the same.
- The optimal cost for the new input is 1.
- The cost of the changed player is at least $1 + \frac{n-1}{n} - \epsilon$.
- The approximation ratio is at least $2 - \frac{1}{n} - \epsilon$.

Outline

Monotone algorithms

Algorithms and Monotonicity

- Monotonicity, which is not specific to the scheduling task problem but it has much wider applicability, poses a new challenging framework for designing algorithms.
- In the traditional theory of algorithms, the algorithm designer could concentrate on how to solve every instance of the problem by itself.
- With monotone algorithms, this is no longer the case. The solutions for one instance must be consistent with the solutions of the remaining instances—they must satisfy the Monotonicity Property.
- Monotone algorithms are holistic algorithms: they must consider the whole space of inputs together.

Some open problems

Scheduling unrelated machines

- Characterize the truthful mechanisms for scheduling unrelated machines.
- Close the gap between 2.41 and n for this problem.
- Improve the bounds 2 and $\Theta(n)$ for randomized mechanisms.
- Study the fractional allocation version.

Combinatorial auctions

- Characterize the truthful mechanisms for more general settings such as the combinatorial auction problem

Outline

Outline

The adwords framework

For a fixed combination of keywords (e.g. “blue lagoon”), we have the following (simplified) situation:

- There are n players/bidders. Each player i has
  - a private value v_i
  - a weight w_i (a “quality metric”, assigned by the search engine).
- There are m ad slots
- For each player i and slot j, let $CTR_{i,j}$ denote the (estimated) probability that the ad of player i in slot j will be clicked.
- Player i has (expected) payoff

$$CTR_{i,j} \cdot v_i$$

The adwords auction

- The players bid values $\hat{v}_1, \ldots, \hat{v}_n$
- The mechanism computes $w_i \cdot \hat{v}_i$ and sorts the players with these values. Let x_i denote the position of player i in this order.
- The mechanism allocates the slots according to this order.
- The players pay $p_1, \ldots, p_n$
- We assume that a player pays nothing when he does not get a slot.

- The utility of player i is $CTR_{i,x_i} \cdot (v_i - p_i)$
- Question: How should we compute the payments?

The adwords auction

Let's reorder the players so that

$$w_1 \cdot \hat{v}_1 \geq w_2 \cdot \hat{v}_2 \geq \cdots \geq w_n \cdot \hat{v}_n$$

Google AdWords asks for payments

$$p_i = \frac{w_{i+1}}{w_i} \hat{v}_{i+1}$$

which is equal to the minimum value the player can declare and still keep the position.
What are the values of w_i?

- Overture/Yahoo! (rank-by-bid) $w_i = 1$
- Google (rank-by-revenue) $w_i = CTR_{i,1}$

Is it truthful?

It looks like the Vickrey (second-price) auction, but is it truthful?

Is it truthful?

It looks like the Vickrey (second-price) auction, but is it truthful?

No!

Example

- Suppose that all $w_i = 1$ and $CTR_{i,j} = 0.1$ are equal.
- There are 2 slots and 3 players with $v_1 = 100$, $v_2 = 80$, $v_3 = 30$.
- The first 2 players get the slots and pay 80 and 30.

Is it truthful?

It looks like the Vickrey (second-price) auction, but is it truthful?

No!

Example

- Suppose that all $w_i = 1$ and $CTR_{i,j} = 0.1$ are equal.
- There are 2 slots and 3 players with $v_1 = 100$, $v_2 = 80$, $v_3 = 30$.
- The first 2 players get the slots and pay 80 and 30.
- But player 1 could gain by declaring 70. Why?

Is it truthful?

It looks like the Vickrey (second-price) auction, but is it truthful?

No!

Example

- Suppose that all $w_i = 1$ and $CTR_{i,j} = 0.1$ are equal.
- There are 2 slots and 3 players with $v_1 = 100$, $v_2 = 80$, $v_3 = 30$.
- The first 2 players get the slots and pay 80 and 30.
- But player 1 could gain by declaring 70. Why?
- He will get the second slot and he will pay 30 (instead of 80). And since $CTR_{1,1} = CTR_{1,2}$, he does not really prefer the first slot.

Is VCG better?

VCG would give the slots to the same players, but the payments should be different:

- The second player should pay again 30
- The first player should pay also 30

The VCG mechanism is truthful. But the search engine gets smaller revenue.

Open problem

Sponsored search differs in two major aspects from this simple model:

- Budgets: Each player has a budget for every day
- Repeated game: The game is played repeatedly with (almost) the same players

Outline

Digital goods auction

- There is an unlimited-supply good (e.g. a song in iTunes) and n potential buyers.
- Each buyer i has a (private) value v_i for the good.
- We want to design a single-round, sealed-bid auction which maximizes profit.
- Players bid $\hat{v}_1, \ldots, \hat{v}_n$ and the auctioneer decides who gets the good and at what price.

Performance measure

- The optimal profit ($\sum_i v_i$) is not possible. What then?
- Compare with single-price optimal profit
- Use competitive analysis (worst-case analysis)

- Problem: Highest bid may be much higher than the rest
- Truthful mechanisms can extract only the second-highest bid (e.g. Vickrey)
- Therefore the competitive ratio is unbounded
- Solution: Compare with the optimal single-price auction which sells at least 2 items
- Reorder the values so that $v_1 \geq \cdots \geq v_n$. We compare with

$$F^{(2)}(v) = max_{i\geq 2} i \cdot v_i$$

Truthful mechanisms

- An auction is **bid-independent** when each player is offered a price that depends on the bids of the other players.
- If the offer exceeds his value he gets the good and pays the price, otherwise he does not get the good and pays nothing.

Theorem

An auction is truthful if and only if it is bid-independent.

Random sampling

Random sampling optimal price (RSOP)

- Solicit bids $v = (v_1, \ldots, v_n)$
- Partition them randomly into two parts v' and v''
- For each part, use the other part as sample to decide the price
- Offer the players of the first part the price which maximizes $F^{(1)}(v'')$ (the optimal price for the second part)
- Similarly for the second part

Random sampling

Example

Suppose we have 2 players with values $(101, 100)$. The optimal profit is $F^{(2)} = 200$.
RSOP does the following:

- With probability $1/2$, both players end up in the same part. The offered price will be $(0, 0)$. The profit of the auction will be 0.
- With probability $1/2$, they end up in different parts. The offered prices will be $(100, 101)$. Only the first player will accept and the profit of the auction is 100.
- The expected profit is $1/2 \cdot 0 + 1/2 \cdot 100 = 50$ and competitive ratio is 4 $(= 200/50)$.

Results

- Deterministic auctions have unbounded competitive ratio
- Randomized lower bound: 2.42
- Randomized upper bound: 3.25
- RSOP is 15-competitive
- Conjecture: RSOP is 4-competitive

Thank you!

References

For a good introduction to these issues see the book "Algorithmic game theory" by Nisan, Roughgarden, Tardos, and Vazirani (available online).