Mechanism design

Lecture 2: Multi-parameter mechanisms as algorithms

Elias Koutsoupias
Oxford

ADFOCS 2024

Multi-parameter mechanism design

A look at multidimensional truthfulness

Recall the characterization of truthfulness for single item auctions / **single parameter domains**.

Theorem

A single-parameter mechanism is truthful if and only if

- *the utility $u(v)$ of the bidder is a convex function of the private value v.*
- *the probability of getting the item is given by*

$$a(v) = u'(v)$$

It generalizes to mutli-item auctions / **multi-parameter domains**:

Theorem

A mechanism is truthful if and only if

- *the utility $u(v)$ is a convex function*
- *the probability of getting item j is given by*

$$a_j(v) = \frac{\partial u(v)}{\partial v_j}$$

Important properties of convex functions

Recall the propositions for single-variable convex functions:

Proposition

For every function g, the function f defined by

$$f(x) = \sup_y \{x \cdot y - g(y)\},$$

is convex.

Proposition

For every convex function f, there exists a function f^* (called the conjugate of f), such that

$$f(x) = \sup_y \{f'(y) \cdot x - f^*(f'(y))\}$$

Both propositions hold for functions of many variables. Simply interpret "$\cdot$" as inner product, and f' as ∇f.

$$f(x) = \sup_y \{\nabla f(y) \cdot x - f^*(\nabla f(y))\}$$

A look at multidimensional truthfulness

Putting everything together. Consider a truthful mechanism (a, p) for an agent with **vector of values v and bid vector b**:

- the **utility** of the agent is

$$u(v) = \sup_b \{a(b) \cdot v - p(b)\},$$

  which is convex even if the mechanism is not truthful.
- Convexity implies

$$u(v) = \sup_y \{\nabla u(v) \cdot v - u^*(\nabla u(y))\}$$

- The **allocation vector** is $a(v) = \nabla u(v)$, and
- the **payment** is $p(v) = u^*(\nabla u(v))$, where u^* is the conjugate of u.

We can add a constant to the payment function without affecting truthfulness: $p(v) = u^*(\nabla u(v)) + \text{const}$.

Take away message for truthfulness

A mechanism (a, p) that produces **legal outcomes** is truthful if for every agent

- the allocation function $a(v)$ is **monotone**, that is, $a(v) = \nabla u(v)$ for some convex function $u(v)$
- the payment function $p(v)$ **depends only** on the allocation $a(v)$ and the values of the other bidders.

Furthermore, if the allocation function $a(v)$ is monotone there exists payment function $p(v)$ for which the mechanism (a, p) is truthful, and vice versa.

Affine maximizers

VCG and affine maximizers

Definition (Affine maximizer)

In an affine maximizer (or generalized VCG) there are constants $\lambda_i \geq 0$ (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)

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

VCG is the special case of $\lambda_i = 1$, $\gamma_j = 0$

VCG and affine maximizers

Theorem

Affine maximizers are truthful.

Why? They align the utility of every bidder with the **weighted social welfare** ($= \sum_i \lambda_i v_{ij} + \gamma_j$): payments are such that **the utility of every agent is maximized when the weighted social welfare is maximized.**

Roberts theorem

Beyond VCG and affine maximizers?

Theorem (Roberts, 1979)

For unrestricted domains with 2 or more players and at least 3 outcomes, the only truthful deterministic mechanisms are the affine maximizers.

- Major open problem: extend Roberts theorem to combinatorial auctions, scheduling, and other domains.

Mechanisms without payments

Theorem (Gibbard-Satterthwaite)

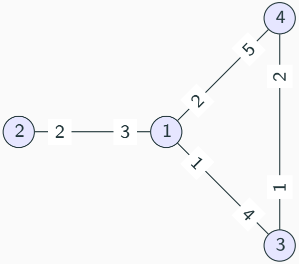
The only truthful mechanisms without money are dictatorships

(In retrospect, a corollary of Roberts theorem.)

- In other words, truthful mechanisms without payments are very weak.
- On the other hand, there are some very interesting mechanisms without payments: For example the stable matching algorithm.

The graph balancing problem

The graph balancing problem



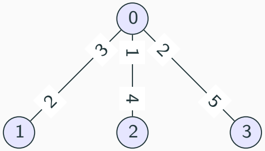
Graph balancing problem: Given a graph in which edges have two weights (one for each of its nodes), orient the edges to **minimize the maximum load** on the nodes. The load of a node is the corresponding weight of all edges oriented towards the node.

Let $t_{i,j}$ denote the weight of node i on edge $\{i,j\}$.

Mechanism design setting: the nodes are agents and their weights are private values. The agents want to minimize the load allocated to them.

Truthful balancing on stars

Consider the balancing setting on stars. The VCG and its generalizations have large approximation ratio.



VCG mechanism (edge-independent)

$$\arg\min_S \sum_{j \in S} t_{0,j} + \sum_{j \notin S} t_{j,0}$$

Approximation ratio: $n - 1$ (= number of leaves)

Affine minimizers / Weighted VCG

$$\arg\min_S \sum_{j \in S} t_{0,j} + \lambda \sum_{j \notin S} t_{j,0}$$

Approximation ratio: $\Theta(\sqrt{n})$, when $\lambda = \Theta(1/\sqrt{n})$

Truthful balancing on stars

A new mechanism:

Hybrid mechanism

$$\arg\min_S \sum_{j \in S} t_{0,j} + \max\nolimits_{j \notin S} t_{j,0}$$

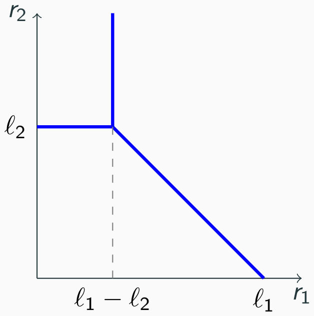
Compare with

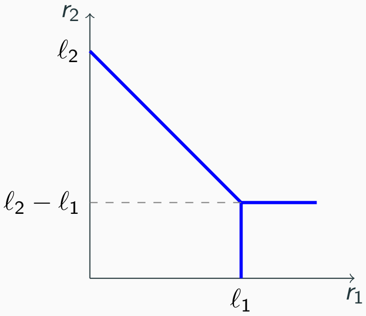
VCG mechanism

$$\arg\min_S \sum_{j \in S} t_{0,j} + \sum\nolimits_{j \notin S} t_{j,0}$$

The geometry of the Hybrid mechanism

The geometry of the Hybrid mechanism (where $r_i = t_{0,i}$, $\ell_i = t_{i,0}$). The figure shows the partition of the space into allocations of the root, when the there are two leaves. For example, the root gets both edges when both r_1 and r_2 are small.





Truthful balancing on stars - upper and lower bound

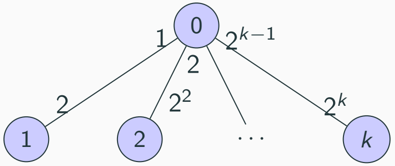
Theorem

The Hybrid mechanism for stars is truthful and has approx ratio 2

Proof of upper bound: Let S be the set allocated to the root. Then

$$\begin{aligned}\text{cost of Hybrid} &\leq \min_S \sum_{j\in S} t_{0,j} + \max_{j\notin S} t_{j,0} \\ &\leq \min_S 2\max\Big\{\sum_{j\in S} t_{0,j}, \max_{j\notin S} t_{j,0}\Big\} = 2\text{OPT}\end{aligned}$$

Lower bound:



What about other graphs?

Definition (Star-Cover mechanism)

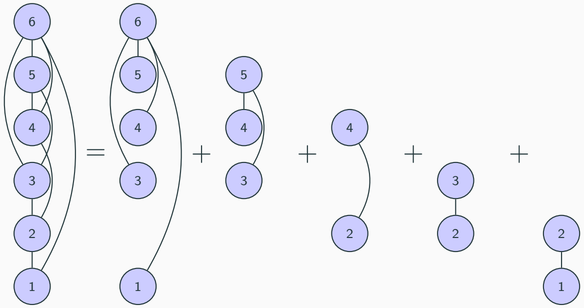
Given an arbitrary graph, partition its edges into stars $T_1, \ldots, T_r$.
Run the Hybrid mechanism for each star independently.

Theorem

Star-Cover is truthful and has approximation ratio $2r$.

Theorem

The approximation ratio of a k-inductive (or k-degenerate) graph is at most $2(k+1)$.



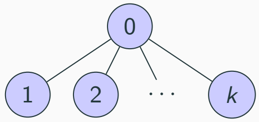
- trees : 4
- planar graphs : 12
- k-trees : $2(k+1)$
- treewidth k : $2(k+1)$

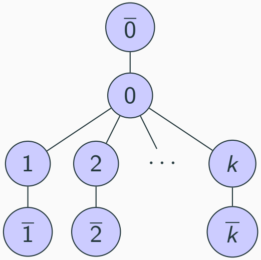
Trees

Theorem

The approximation ratio for trees is in $[2.61, 4]$.

- Upper bound: trees are 1-inductive
- Lower bound: expand the lower bound for stars.





Extensions to L^p-norm

L^p-norm objective: minimize or maximize $(\sum_i \mathrm{COST}_i^p)^{1/p}$.

- $p = \infty$ (minimize the makespan)
- $p = 1$ (total welfare - VCG is optimal)
- $p \to 0$ (Nash Social Welfare)

The Hybrid mechanism can be extended to this objective.

Theorem

For the problem of minimizing the L^p-norm, the Hybrid L^p mechanism for stars has approximation ratio of most $2^{|p-1|/p}$.

This is optimal, for values of $p \geq 1$.

The unrelated machines scheduling problem

The scheduling problem

Input

- n machines (aka bidders, agents, players)
- m tasks

$$\begin{bmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,m} \\ \vdots & \vdots & & \vdots \\ t_{i,1} & t_{i,2} & \cdots & t_{i,m} \\ \vdots & \vdots & & \vdots \\ t_{n,1} & t_{n,2} & \cdots & t_{n,m} \end{bmatrix}$$

$t_{i,j}$: execution of time of task j by machine i

Output

Allocation $a_{i,j}$: 1 if and only if machine i gets task j

Objective

Minimize the makespan $= \max_i \sum_j a_{i,j} t_{i,j}$

The scheduling problem

In the classical algorithmic setting:

- NP-hard to approximate within a factor of 1.5
- There exists a polytime 2-approximation algorithm (Lenstra-Smoys-Tardos, 1987)
- Major open problem to close the gap $[1.5, 2]$

Mechanism design setting

$$\begin{bmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,m} \\ \vdots & \vdots & & \vdots \\ t_{i,1} & t_{i,2} & \cdots & t_{i,m} \\ \vdots & \vdots & & \vdots \\ t_{n,1} & t_{n,2} & \cdots & t_{n,m} \end{bmatrix}$$

The execution times $t_i = (t_{i,1}, \dots, t_{i,m})$ of machine i are **private values**.

To incentivize the machines to be truthful (i.e., to report their actual private values), the mechanisms pays them.

- A mechanism is defined by the allocation function $a(t)$ and the payment functions $(p_1(t), \dots, p_n(t))$.
- Each machine i reports values t_i^*, perhaps different than t_i, that maximize its utility

Example: the VCG mechanism

The Vickrey-Clarke-Groves (VCG) mechanism

- allocates each task, independently of the others
- it gives each task j to the machine with the minimum (declared) value $t_{i,j}$ and pays the second minimum value.

The VCG mechanism has approximation ratio exactly n. Lower bound:

$$\begin{bmatrix} \mathbf{1}-\epsilon & \mathbf{1}-\epsilon & \cdots & \mathbf{1}-\epsilon \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

Nisan and Ronen conjectured that no mechanism has better approximation ratio.

The Nisan-Ronen conjecture

Theorem [Christodoulou-Koutsoupias-Kovacs, STOC 2023]

No deterministic mechanism can achieve an approximation ratio less than n for the scheduling problem of n unrelated machines.

Conjectured by Noam Nisan and Amir Ronen [STOC 1999].

Why focus on scheduling?

- scheduling is an important classical optimization problem
- the objective is the maximum of completion times
    - in contrast, the sum of completion times can be solved optimally by VCG
    - similarly, Minimum Spanning Tree and TSP admit "easy" mechanisms
- multi-parameter domain
    - in contrast, single parameter domains are usually easy
- extremely rich and challenging setting

Monotone algorithms for multiple tasks

Monotonicity for multiple tasks

Multiple parameter monotonicity / truthfulness

If $a_{i,j}(t)$ is the probability of allocating task j to machine i, then

- $a_i(t) = (a_{i,1}(t), \ldots, a_{i,m}(t))$ is monotone in (vector) t_i, i.e., when we change t_i to t'_i:

$$(\, a_i(t') - a_i(t)\,) \cdot (t'_i - t_i) \leq 0$$

  ("$\cdot$" means the inner product of vectors)

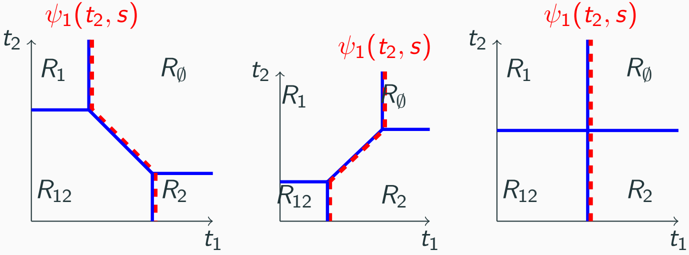
- $a_i(t)$ is the gradient of a concave function (= the dis-utility of the player)

The types of 2×2 deterministic monotone allocations

Consider 2 machines and 2 tasks, with input $\begin{pmatrix} t_1 & t_2 \\ s_1 & s_2 \end{pmatrix}$.

An algorithm partitions the space of the first machine into 4 parts, one for each possible allocation.

Monotonicity, $\Delta a_i \cdot \Delta t_i \leq 0$, implies that the partition must have the following special partitions (called **quasi-bundling, quasi-flipping, and task-independent**, respectively).



$\psi_1(t_2, s)$ is the boundary function for task 1.

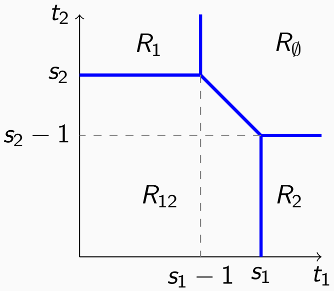
Relaxed affine minimizers

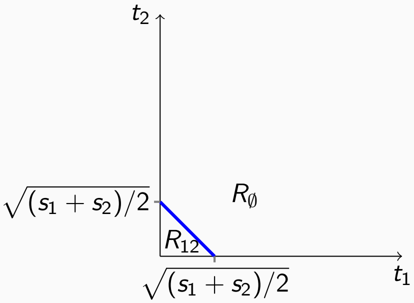
Affine minimizers have boundaries that are affine functions, e.g., $\psi_1(s_1) = \lambda\, s_1 + \gamma$, for some constants λ and γ.

Relaxed affine minimizers extend affine minimizers as follows: when there are only two possible allocations, the boundaries need only be monotone.

Relaxed affine minimizers are truthful mechanisms.

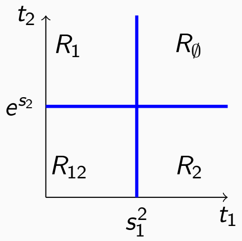
Example of a relaxed affine minimizer:





Relaxed task independent algorithms

Task independent mechanisms are also truthful mechanisms (another generalization of VCG).



Boundaries are independent, except on countably many points. Boundary functions can be arbitrary monotone functions, e.g., $\psi_1(s_1) = s_1^2$.

Characterization of (2 machines, 2 tasks) monotone algorithms

Theorem

There are only two types of monotone deterministic algorithms for 2 machines and 2 tasks:

- *relaxed affine minimizers*
- *relaxed task independent algorithms*

Major open question: what are the possible monotone algorithms for multiple machines and tasks? For randomized algorithms?

Multigraph balancing

Multigraph balancing is a special case of scheduling

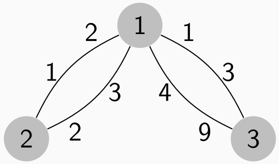
We don't understand the interplay of monotonicity for more than two machines.

To overcome this, we consider a special case of the scheduling problem in which every task can be allocated to at most two machines.

This is equivalent to the multigraph balancing problem: given a multigraph with double weights on its edges (one for each node), orient its edges to minimize the maximum incoming weight.

Multigraph balancing is a special case of scheduling

2	3	4	1
1	2	∞	∞
∞	∞	9	3



Machines $\leftrightarrow$ *Nodes*

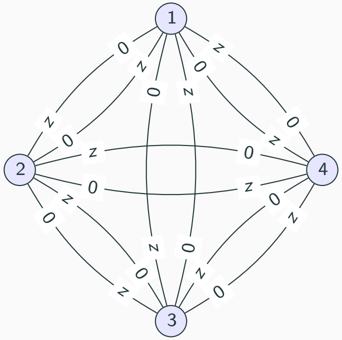
Tasks $\leftrightarrow$ *Edges*

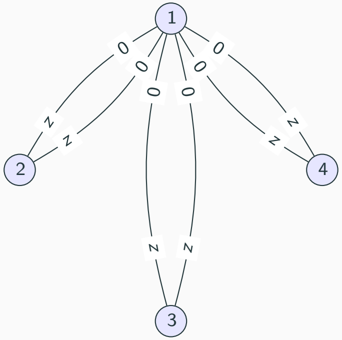
The execution time for the other machines is so high (∞) that no algorithm with poly(n) approximation ratio uses them.

Instances used in the proof of the Nisan-Ronen conjecture: multi-cliques and multi-stars

Instances of graphs that we consider

Multi-cliques and multi-stars with very high multiplicity.





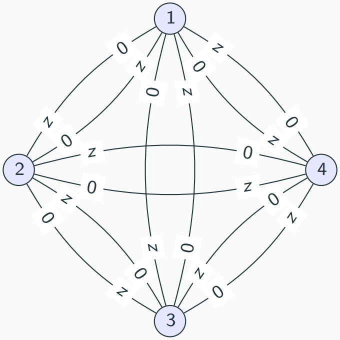
- Every edge $e = \{i, j\}$ has two values: 0 and $z_{i,j} \geq 0$.
- Important property: the optimal allocation has cost 0.

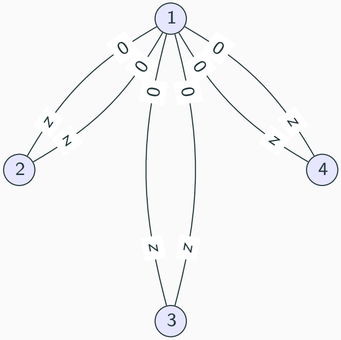
Theorem (Main)

No deterministic monotone algorithm can have an approximation ratio less than n on multi-cliques.

Instances of graphs that we consider

Multi-cliques and multi-stars with very high multiplicity.





- Every edge $e = \{i, j\}$ has two values: 0 and $z_{i,j} \geq 0$.
- Important property: the optimal allocation has cost 0.

Theorem (Main)

No deterministic monotone algorithm can have an approximation ratio less than n on multi-cliques.

High level proof

Towards a contradiction, assume that the approximation ratio is less than n. Then

Theorem (Box theorem)

Every multi-star with sufficiently high multiplicity contains a "box".

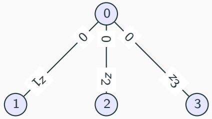
box: a star whose edges are oriented almost independently

Theorem (Nice multi-star theorem)

Every multi-clique with sufficiently high multiplicity contains a "nice" multi-star of a given multiplicity.

nice multi-star: if it contains a box of size n, the approximation ratio is n.

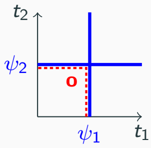
Box theorem

Consider the star 

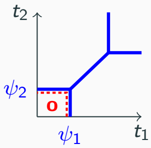
- Fix the values of the leaves
- $\psi_j(z_j) =$ maximum (supremum) value for which the root takes task j, when we keep all other tasks at value 0.
- What will happen if we increase the values of every task j to $\psi_j(z_j) - \epsilon$?
- The star is called a **box** if the root will still take all the tasks.

$$\begin{bmatrix} 0 & 0 & \dots & 0 & \dots & 0 \\ z & & & & & \\ & z & & & & \\ & & \ddots & & & \\ & & & z & & \\ & & & & \ddots & \\ & & & & & z \end{bmatrix} \rightarrow \begin{bmatrix} \psi_1(z) & \psi_2(z) & \dots & \psi_j(z) & \dots & \psi_{n-1}(z) \\ z & & & & & \\ & z & & & & \\ & & \ddots & & & \\ & & & z & & \\ & & & & \ddots & \\ & & & & & z \end{bmatrix}$$

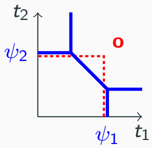
Examples of box and non-box stars for 2 and 3 leaves:



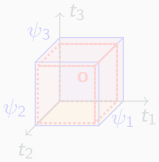
box



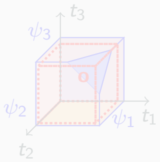
box



non-box

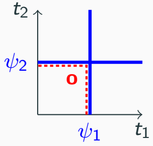


box

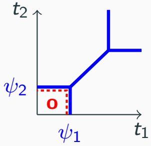


non-box

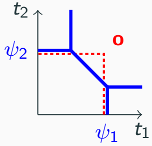
Examples of box and non-box stars for 2 and 3 leaves:



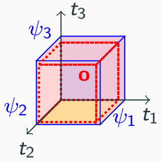
box



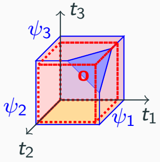
box



non-box



box



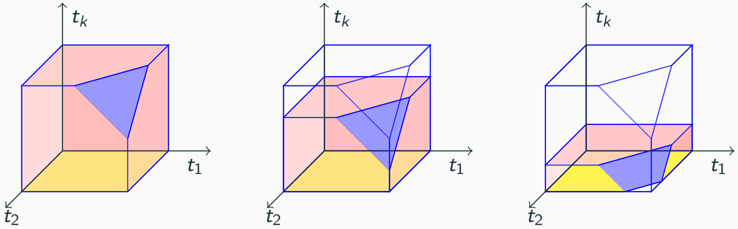
non-box

Theorem (Box theorem)

Every multi-star with sufficiently high multiplicity contains a box.

Proof (intuition):

- induction on the number of machines $k = 2, \ldots, n$;
- induction step $(k-1) \to k$: assume $\{1, 2, \ldots, k\}$ is not a box, but its sides are boxes



- $\psi_k(s_k)$ cannot be linear in s_k because by changing s_k the shape would move rectilinearly (as above), which leads to a contradiction.
- By the 2x2 characterization, the allocation of task k is independent of $t_{k'}$ of every parallel task k', so $\{1, 2, \ldots, k'\}$ is a box.

Approximation ratio of multi-stars is $\Theta(\sqrt{n})$

A corollary of the box theorem is

Theorem

The approximation ratio of multi-stars is $\sqrt{n-1}$.

Why? Fix all leaf values to z.

- Case 1: $\psi_j(z) \leq z/\sqrt{n-1}$
  if root has value $\psi_j(z) + \epsilon$, task j will be allocated to the leaf, which gives an approximation ratio at least $\sqrt{n-1}$
- Case 2: $\sum_j \psi_j(z) \geq z\sqrt{n-1}$
  root with values $\psi_j(z) - \epsilon$ takes all tasks, which gives again an approximation ratio of at least $\sqrt{n-1}$.

To improve the ratio to n, we need to consider multi-cliques. Our aim is to find a box that satisfies $\sum_j \psi_j(z) \geq (n-1)z$, for some $z > 0$.

Nice multi-star theorem

Nice multi-stars

To get a better approximation ratio, we need to consider multicliques, instead of multistars.

The idea is that a multiclique contains a special type of multistars, called **nice**, with an approximation ratio of n.

Nice multi-stars

What is a nice multi-star $S(q, z)$? A multi-star of $n - 1$ leaves with

- multiplicity q
- every edge has values $(0, z)$
- every simple star satisfies $\sum_i \psi_i(z) \geq (n-1)z$.

Theorem (Nice multi-star theorem)

For every q, there is a multi-clique that contains a nice multi-star $S(q, z)$ for some $z > 0$ — otherwise the approximation ratio is at least n.

Why? A random multi-clique of sufficiently high multiplicity contains a nice multi-star $S(q, z)$, for some z, with positive probability.

By random: each edge has random values $(0, x)$ or $(x, 0)$ for $x \in \{\epsilon, 2\epsilon, \ldots, 1\}$.

Why are there nice multi-stars (with $\sum_i \psi_i(z) \geq (n-1)z$)?

Because on "average" every star in a clique is nice. This is due to Young's inequality.

Lemma (Young's inequality)

If ψ_i and ψ_j are the boundary functions of an edge, then for every $a \geq 0$:

$$\int_0^a \psi_i(x)\,dx + \int_0^a \psi_j(x)\,dx \geq a^2.$$

Why? Because ψ_i and ψ_j are inverse functions.

Why are there nice multi-stars (with $\sum_i \psi_i(z) \geq (n-1)z$)?

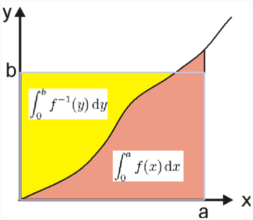
Because on "average" every star in a clique is nice. This is due to Young's inequality.

Lemma (Young's inequality)

If ψ_i and ψ_j are the boundary functions of an edge, then for every $a \geq 0$:

$$\int_0^a \psi_i(x)\,dx + \int_0^a \psi_j(x)\,dx \geq a^2.$$

Why? Because ψ_i and ψ_j are inverse functions.


y
b
$\int_0^b f^{-1}(y)\,\mathrm{d}y$
$\int_0^a f(x)\,\mathrm{d}x$
a
x

On “average” every clique has a nice star

Lemma (Young’s inequality)

If ψ_i and ψ_j are the boundary functions of an edge, then for every $a \geq 0$:

$$\int_0^a \psi_i(x)\,dx + \int_0^a \psi_j(x)\,dx \geq a^2.$$

If ψ’s were **independent**, then on “average” there exists a star such that

$$\int_0^a \sum_i \psi_i(x)\,dx \geq \frac{n-1}{2}a^2$$

$$\exists x : \sum_i \psi_i(x) \geq (n-1)x,$$

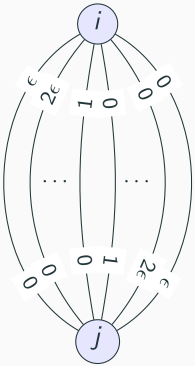
Problems to solve:

- ψ's are not independent, but we need to show that the same x works for all edges
- to be able to apply the Box theorem, we need a nice multi-star of high multiplicity, not a simple star

Fixing the details

- ψ's are not independent, but we need to show that the same x works for all edges
- we need a nice multi-star of high multiplicity, not a simple star

The solution to the first problem : with positive probability there exists a dipole in which all edges have (almost) the same ψ. This requires very high multiplicity.



Fixing the details

- ψ's are not independent, but we need to show that the same x works for all edges
- we need a nice multi-star of high multiplicity, not a simple star

One solution to the second problem comes from extremal hypergraph theory. What we need is a generalization of the Zarankiewicz problem:

Lemma (Zarankiewicz)

For every m and d, there exists k such that every $k \times k$ bipartite graph with density d contains a complete $m \times m$ bipartite graph.

Take away message

Theorem

No deterministic mechanism can achieve an approximation ratio less than n for the scheduling problem of n unrelated machines.

Conclusion

The Nisan-Ronen conjecture is correct : **truthfulness/monotonicity imposes severe restrictions**

What can we do?

- Understand **randomized mechanisms**. They may be more powerful
- Move **beyond DSIC mechanisms**

Randomized / fractional mechanisms

The fractional version of scheduling

Fractional allocations

- With fractional allocations, each task can be split across the machines.
- Computationally easier than the deterministic version: solvable in polynomial time by linear programming.
- fractional approximation ratio $\leq$ randomized approximation ratio

Fractional version: upper bound

Definition

The SQUARE mechanism allocates each task independently. Machine i gets task j with probability inversely proportional to $t_{i,j}^2$.

Theorem

The SQUARE mechanism is truthful and has approximation ratio $(n+1)/2$*.*

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 value from 0 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 $1 + \frac{n-1}{n} - \epsilon$.
- Therefore the approximation ratio is at least $2 - \frac{1}{n} - \epsilon$.

Fractional version: lower bound

A bad input

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

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

- Find the player who gets the largest fraction of the last task and raise its diagonal value from 0 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 $1 + \frac{n-1}{n} - \epsilon$.
- Therefore the approximation ratio is at least $2 - \frac{1}{n} - \epsilon$.

Open problems

Open questions

- What is the power of fractional and randomized monotone algorithms?
- What is the approximation ratio for simple (multiplicity 1) cliques? For trees?
- Understand the communication complexity of truthful combinatorial auctions

Thank you!

Open questions

- What is the power of fractional and randomized monotone algorithms?
- What is the approximation ratio for simple (multiplicity 1) cliques? For trees?
- Understand the communication complexity of truthful combinatorial auctions

Thank you!