Pure equilibria: Existence and inefficiency & Online Auction

Nguyen Kim Thang

Ecole Polytechnique
June 24th, 2009

Rational behaviors

Rational behaviors

* What route to go to work?

Rational behaviors

* What route to go to work?

* Where to open a new competitive facility in Paris?

Rational behaviors

* What route to go to work?

* Where to open a new competitive facility in Paris?

* On Wednesday, what time to have lunch in Polytechnique?

Game Theory + Algorithms

* Entities in society, each with its own information and interests, behave in rational manners.

* Game theory is a deep theory studying such interactions (in economics, political science, ... etc).

* Theoretical computer science studies optimization problems, seeks to optimum, efficient computing, impossibility results, ... etc

Algorithmic Game Theory

* Research field on the interface of game theory and theoretical computer science (mostly algorithms)

* Formulating novel goals and problems, fresh looks on different issues (inspired by Internet, ...).

* The field has phenomenally exploded with many branches: computing Nash equilibrium, mechanism design, inefficiency of equilibria, ... etc

Motivation

* Pure equilibria: existence and inefficiency.

* Online Mechanism Design (Online Auction inspired by Google, Yahoo! Adwords, ...).

* Inspired by real problems.

* Mathematically beautiful.

Outline

- Voronoi Games on graphs
  - NP-complete whether there exists an equilibrium
  - Social cost discrepancy
- Scheduling Games in the Dark
  - Existence of equilibria
  - Optimal non-clairvoyant policy
- Online Algorithmic Mechanism Design
  - Truthful online auction with single-minded bidders

Voronoi Games on Graphs

Voronoi Games

- Summer holiday is also competition season.
- How to make this man happy?

Voronoi Games

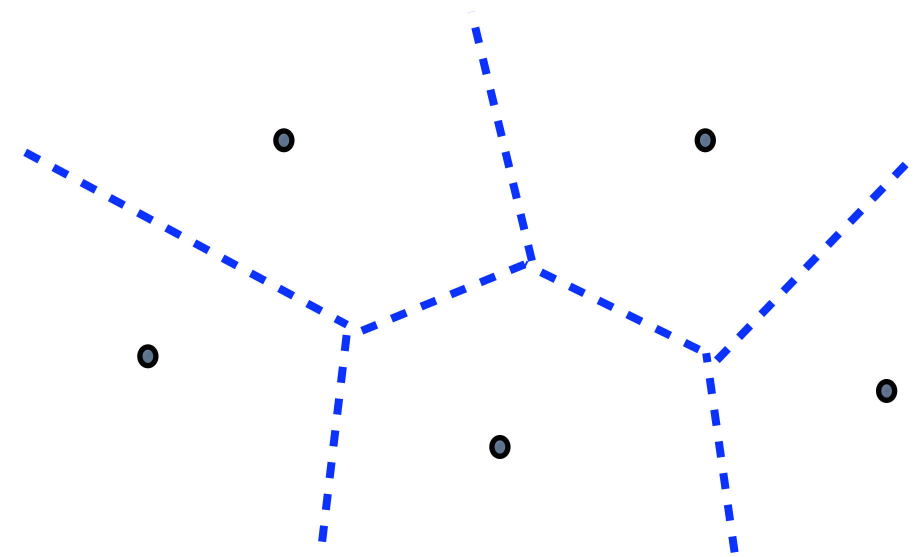
- Summer holiday is also competition season.

- How to make this man happy?

- Application: locations of supermarkets, Internet or mobile phone providers, ...

Voronoi Games

- Summer holiday is also competition season.
- How to make this man happy?



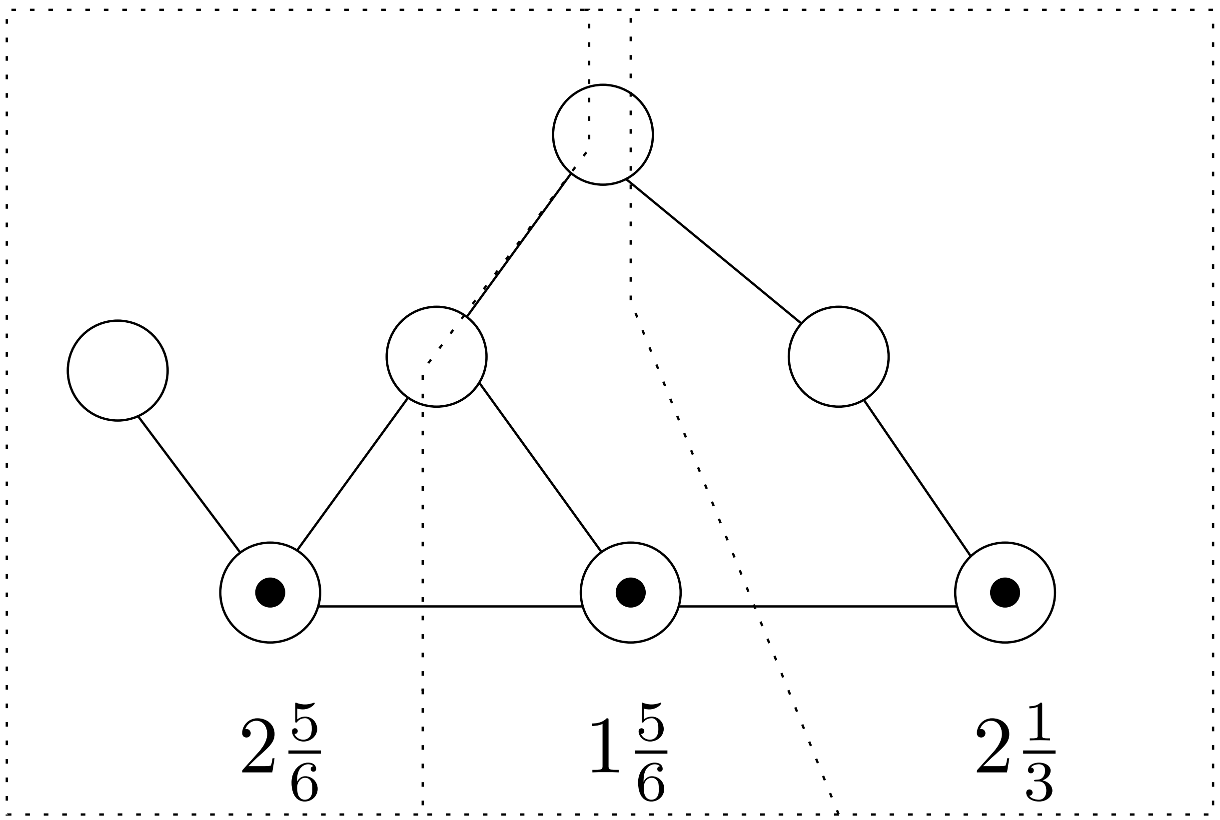
Voronoi Games on graphs

- Given $G(V, E), k$ players whose strategy set is V

- A vertex (client) is assigned in equal fraction to the closest players

- Utility = fractional amount of vertices assigned to the player.

- Social cost = sum of distances over all vertices to the closest player. (k-median optimization problem)



Equilibrium and Complexity

* Equilibrium: strategy profile that is resilient to deviation of each player.

Equilibrium and Complexity

* Equilibrium: strategy profile that is resilient to deviation of each player.

Mixed equilibrium

Pure equilibrium

Equilibrium and Complexity

* Equilibrium: strategy profile that is resilient to deviation of each player.

Mixed equilibrium

choose a distribution over strategies

Pure equilibrium

Equilibrium and Complexity

* Equilibrium: strategy profile that is resilient to deviation of each player.

Mixed equilibrium

Pure equilibrium

deterministically choose a strategy

Equilibrium and Complexity

* Equilibrium: strategy profile that is resilient to deviation of each player.

Mixed equilibrium

always exists (by Nash)

Pure equilibrium

Equilibrium and Complexity

* Equilibrium: strategy profile that is resilient to deviation of each player.

Mixed equilibrium

always exists (by Nash)

Finding: PPAD-complete

Pure equilibrium

Equilibrium and Complexity

* Equilibrium: strategy profile that is resilient to deviation of each player.

Mixed equilibrium

always exists (by Nash)

Finding: PPAD-complete

Pure equilibrium

Finding: PLS-complete

Equilibrium and Complexity

* Equilibrium: strategy profile that is resilient to deviation of each player.

Mixed equilibrium

always exists (by Nash)

Finding: PPAD-complete

Pure equilibrium

Finding: PLS-complete

Existence: NP-hardness

Equilibrium and Complexity

* Equilibrium: strategy profile that is resilient to deviation of each player.

Mixed equilibrium

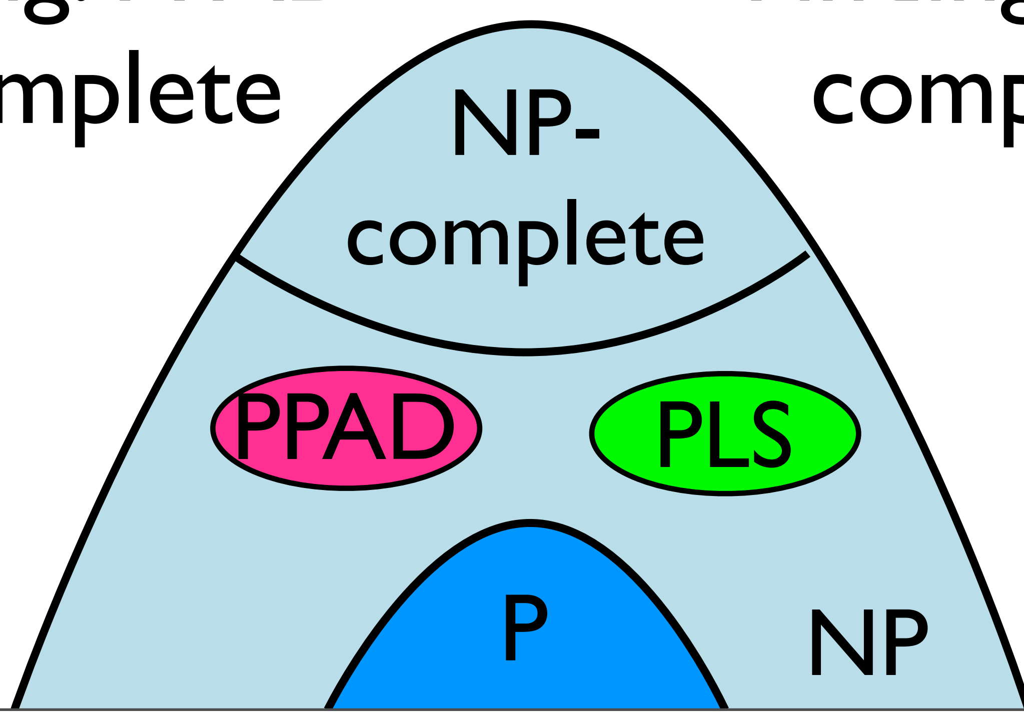
always exists (by Nash)

Finding: PPAD-complete

Pure equilibrium

Finding: PLS-complete

Existence: NP-hardness



Framework in proving NP-hardness

Negated gadget for property *P* of a game + A larger game which encodes a NP-hard problem

→ NP-hardness in deciding whether a game possesses property *P*

Framework in proving NP-hardness

"counter example"

Negated gadget for property *P* of a game

+

A larger game which encodes a NP-hard problem

→ NP-hardness in deciding whether a game possesses property *P*

Framework in proving NP-hardness

"counter example"

Negated gadget for property *P* of a game + A larger game which encodes a NP-hard problem

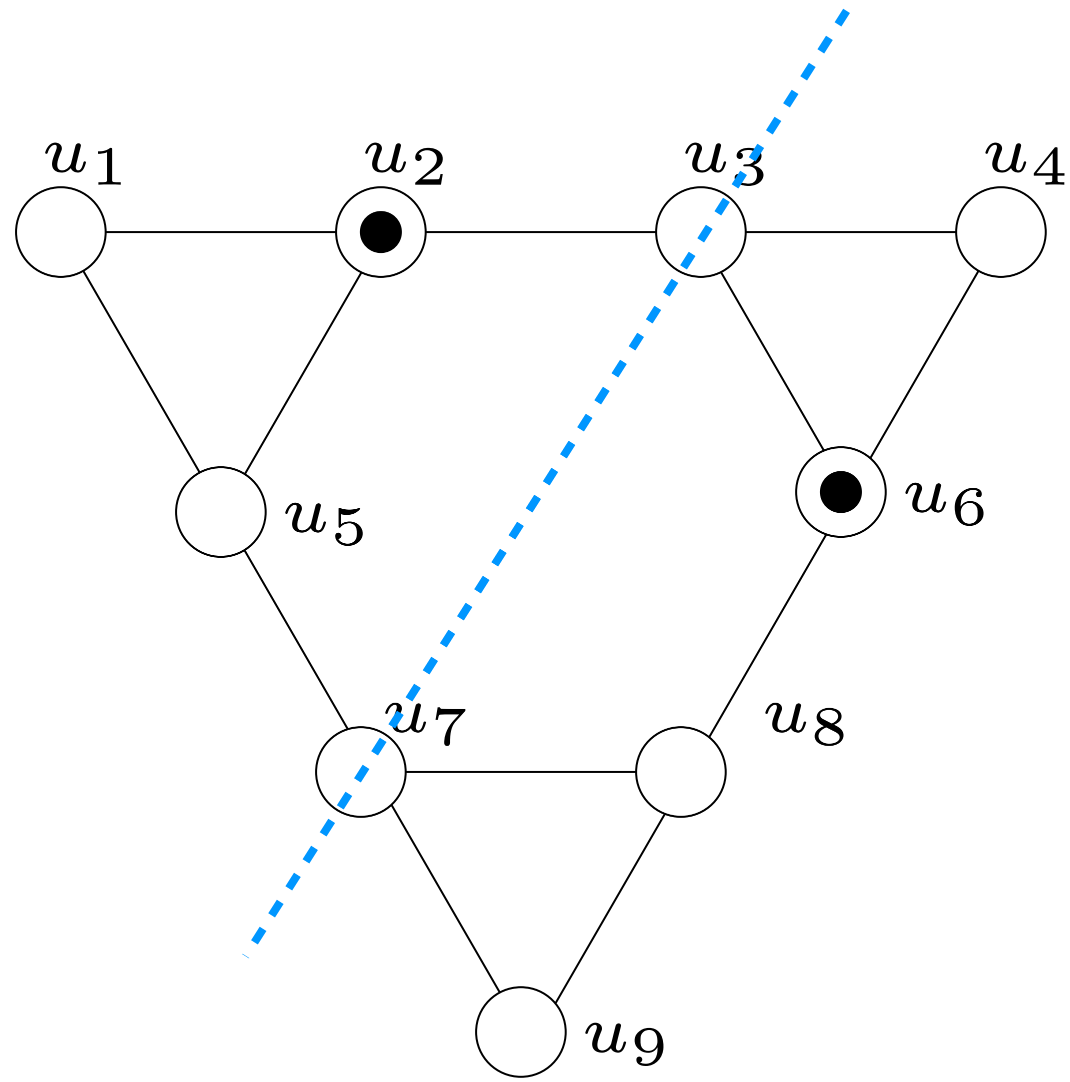
→ NP-hardness in deciding whether a game possesses property *P*

- Voronoi Games
- Matrix Scheduling Games
- Connection Games

Gadget

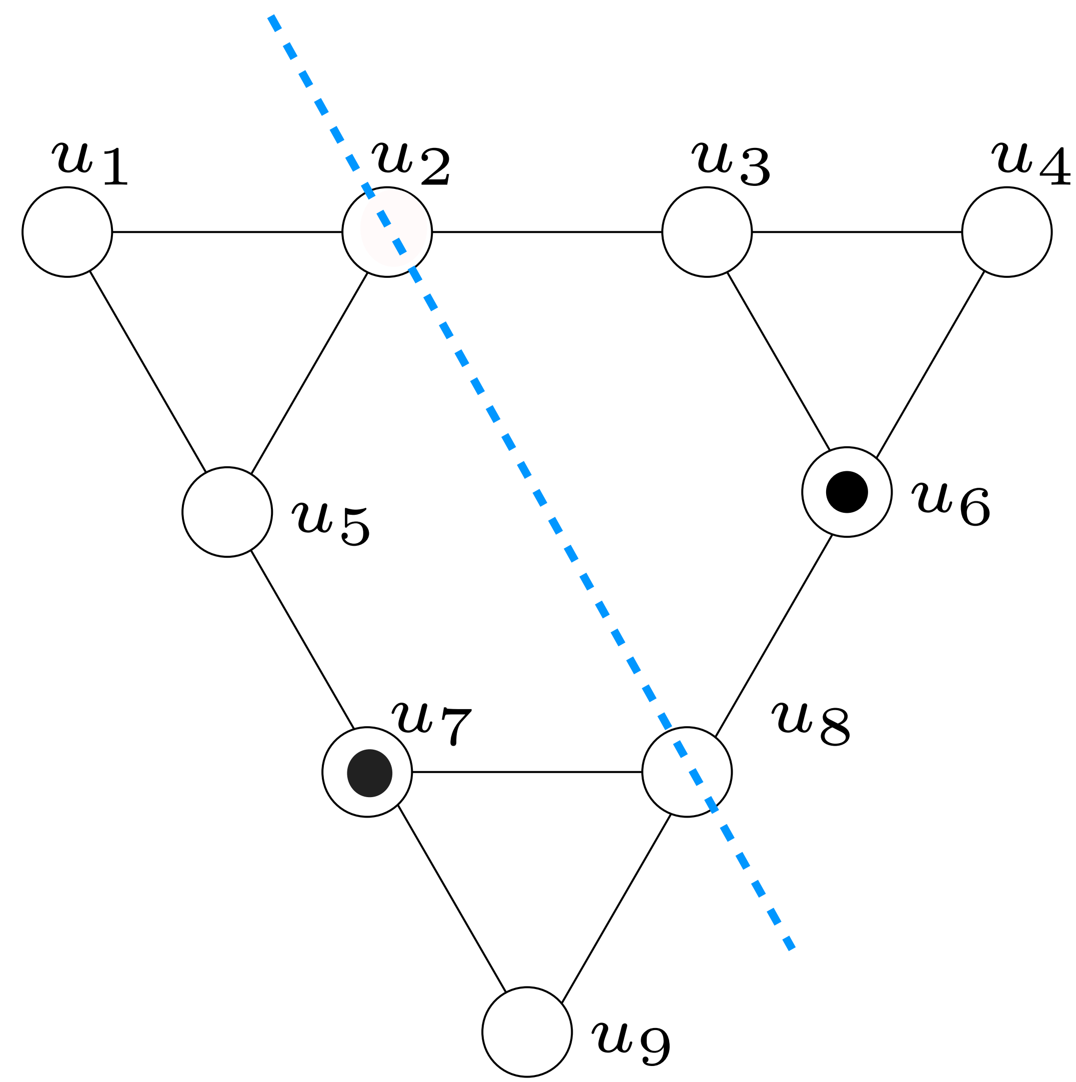
Lemma: There is no Nash equilibrium with 2 players.

Proof: By sym., the first player choose u_2.
Then the second player moves to u_6 and gains 5.
Now the first player can move to u_7 to increase his utility.



Gadget

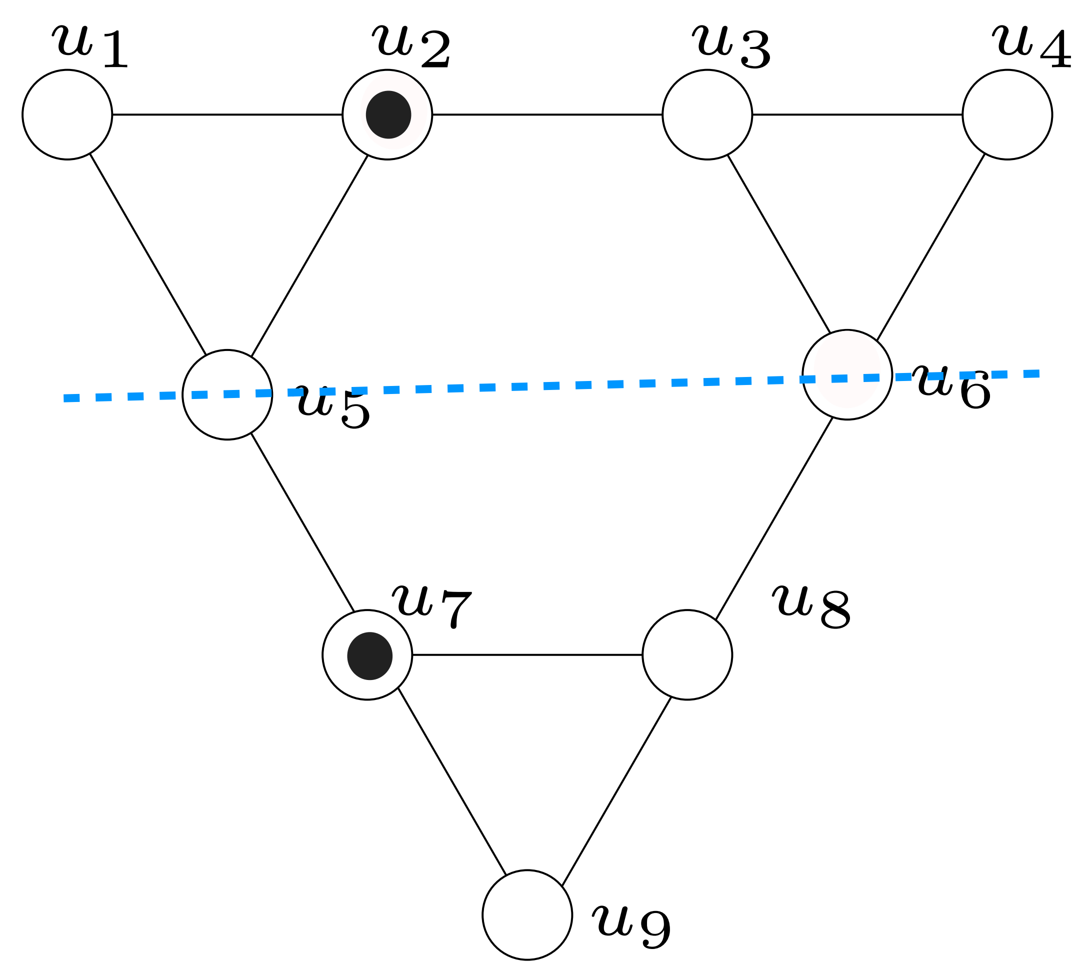
Lemma: There is no Nash equilibrium with 2 players.



Proof: By sym., the first player choose u_2.
Then the second player moves to u_6 and gains 5.
Now the first player can move to u_7 to increase his utility.

Gadget

Lemma: There is no Nash equilibrium with 2 players.

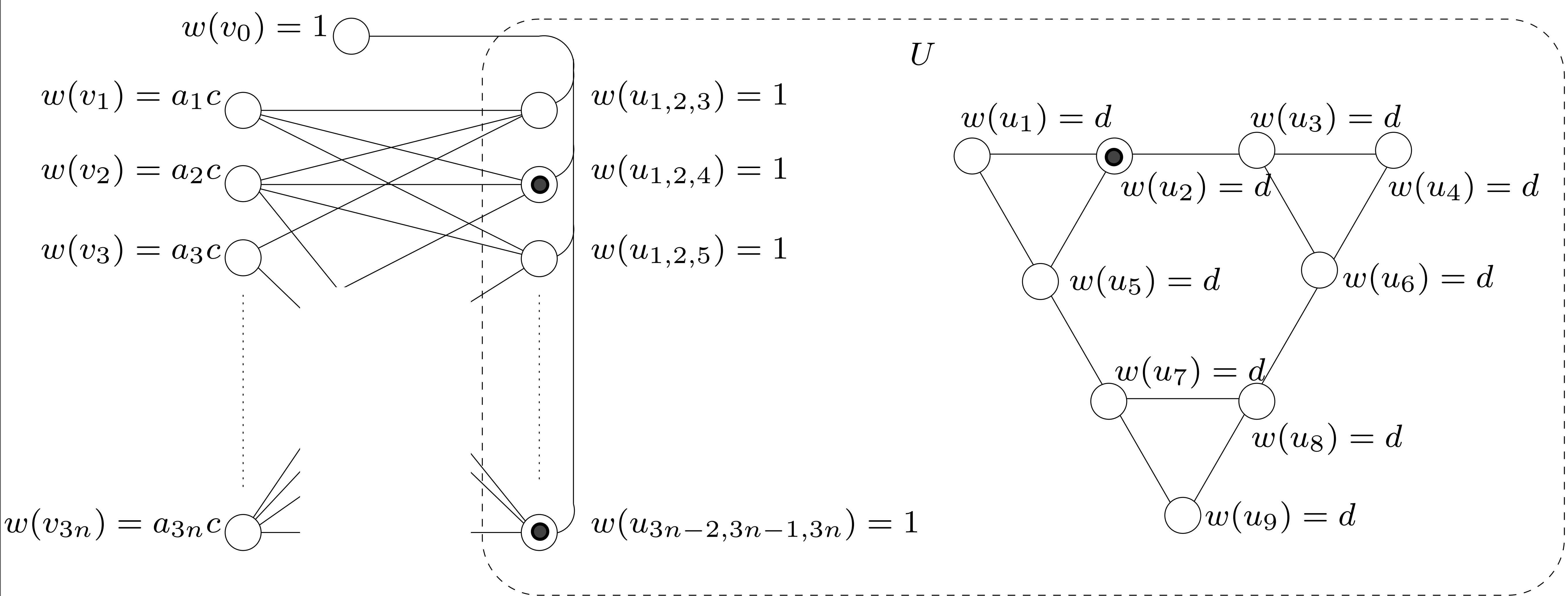


Proof: By sym., the first player choose u_2.
Then the second player moves to u_6 and gains 5.
Now the first player can move to u_7 to increase his utility.

NP-hardness

Theorem: It is NP-hard to decide whether a Voronoi game admits an equilibrium.

Proof (high-level):

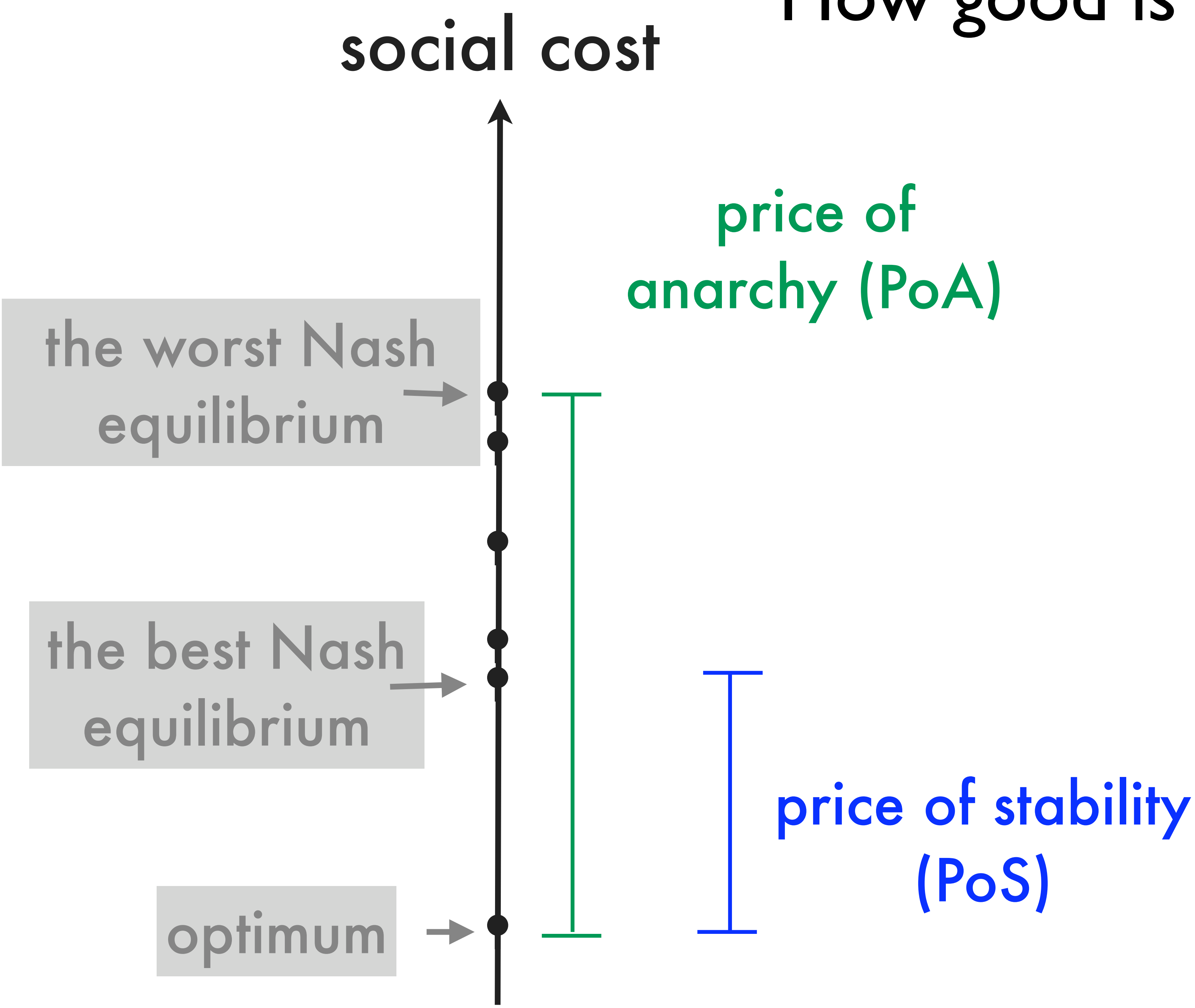


Inefficiency

How good is an equilibrium ?

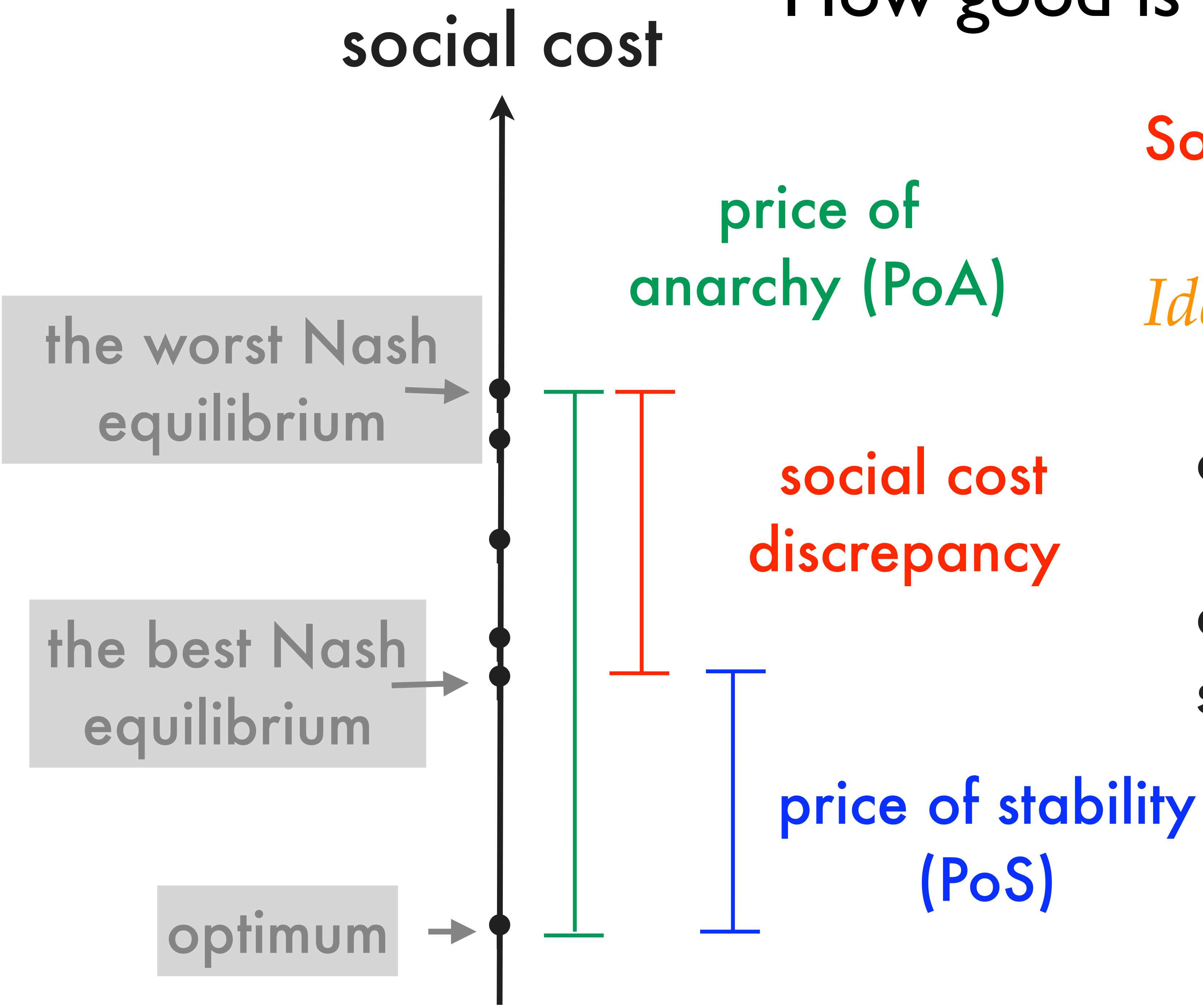
Inefficiency

How good is an equilibrium ?



Inefficiency

How good is an equilibrium ?



Social cost discrepancy:
worst Nash / best Nash

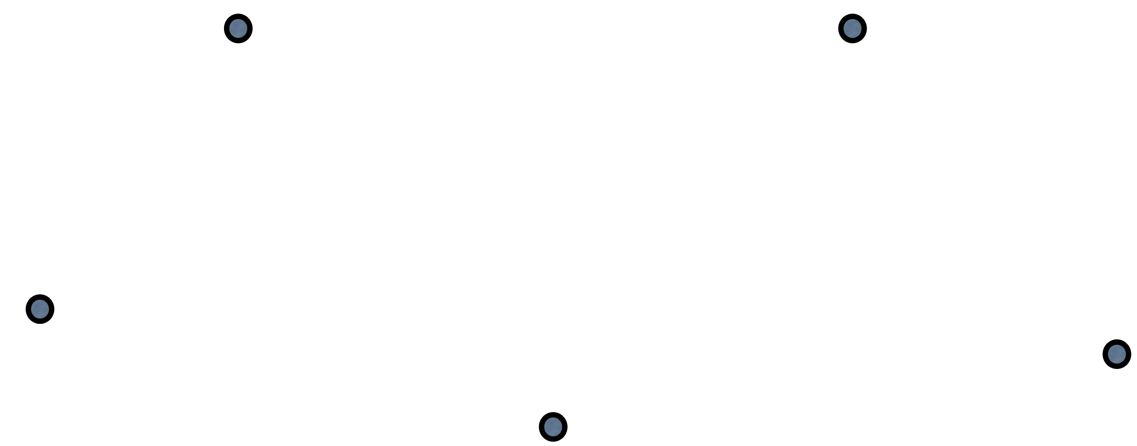
Ideas:

- Measure the degree of choice in a game.
- Unfair to compare the cost with OPT in selfish setting.

Delaunay triangulation

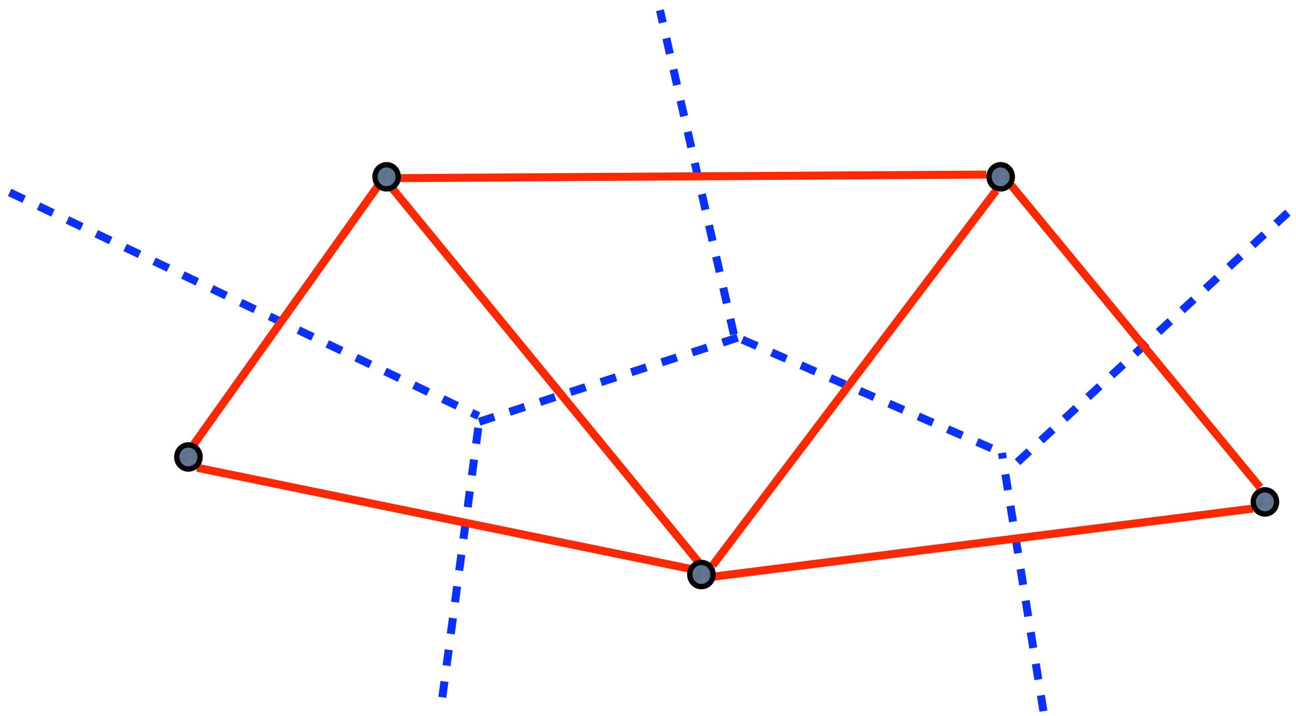
Delaunay triangulation

- Delaunay triangulation:



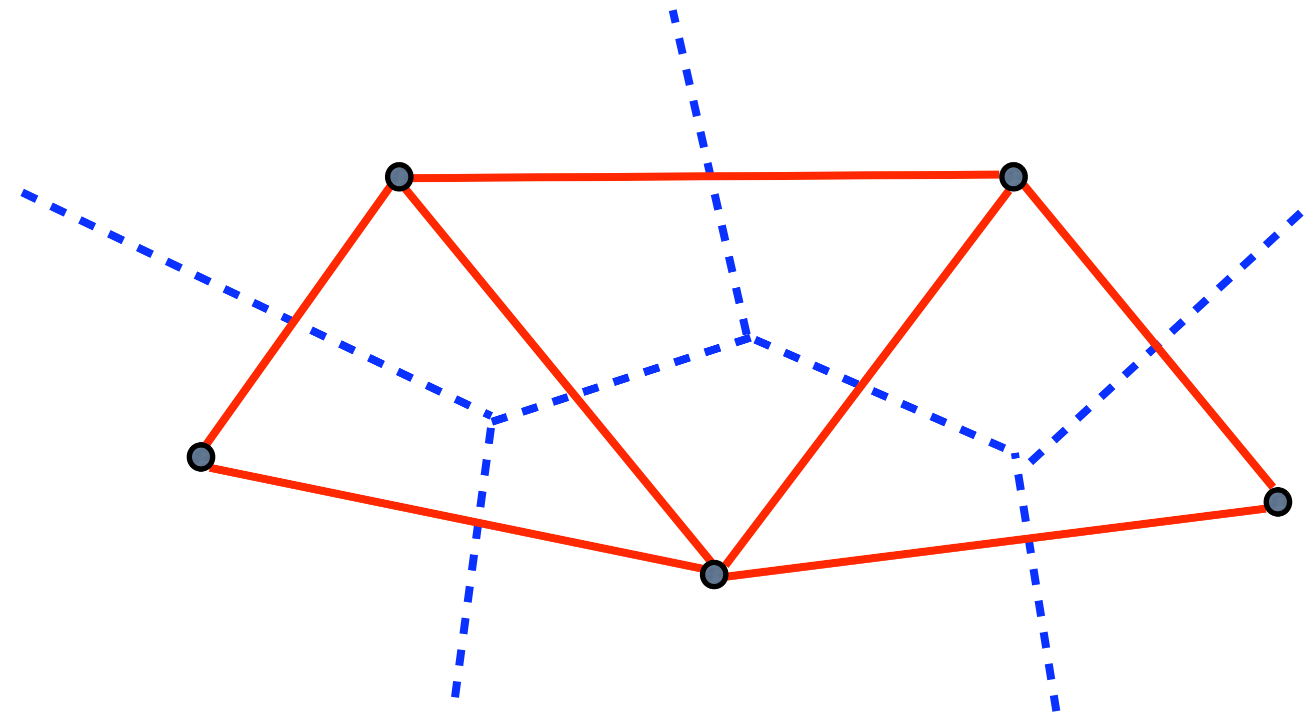
Delaunay triangulation

* Delaunay triangulation:

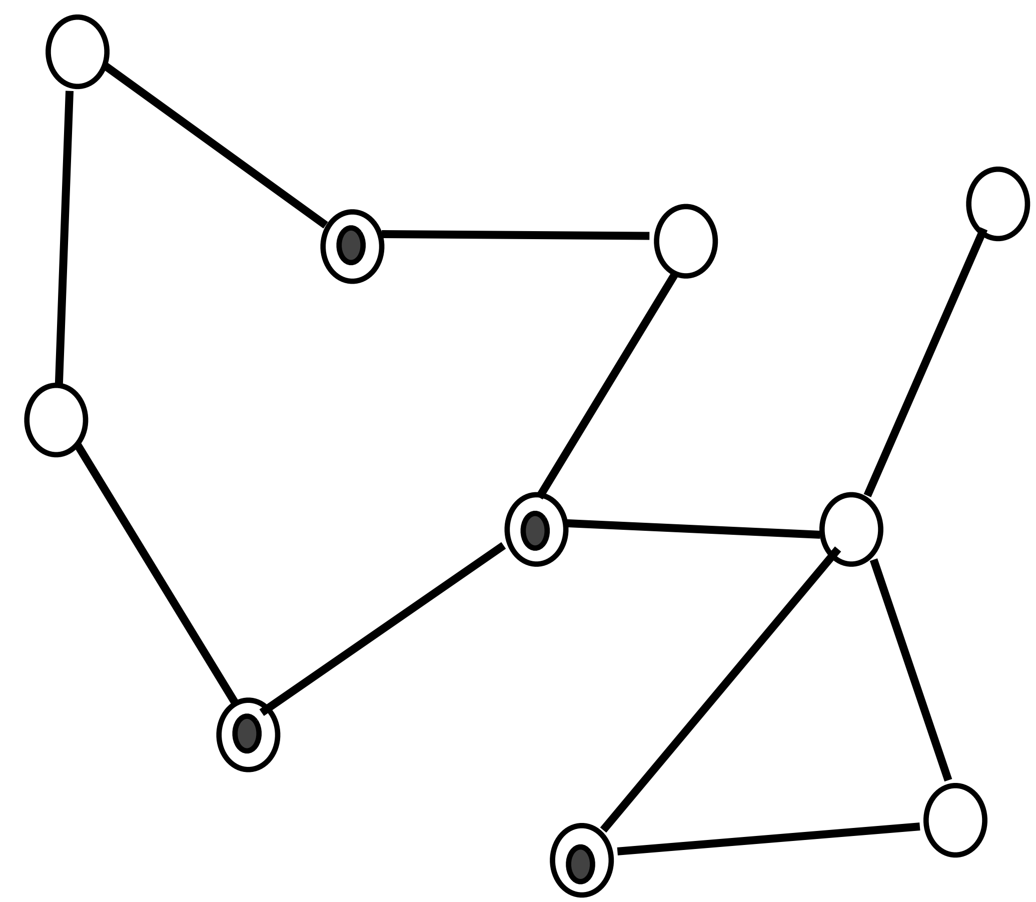


Delaunay triangulation

* Delaunay triangulation:

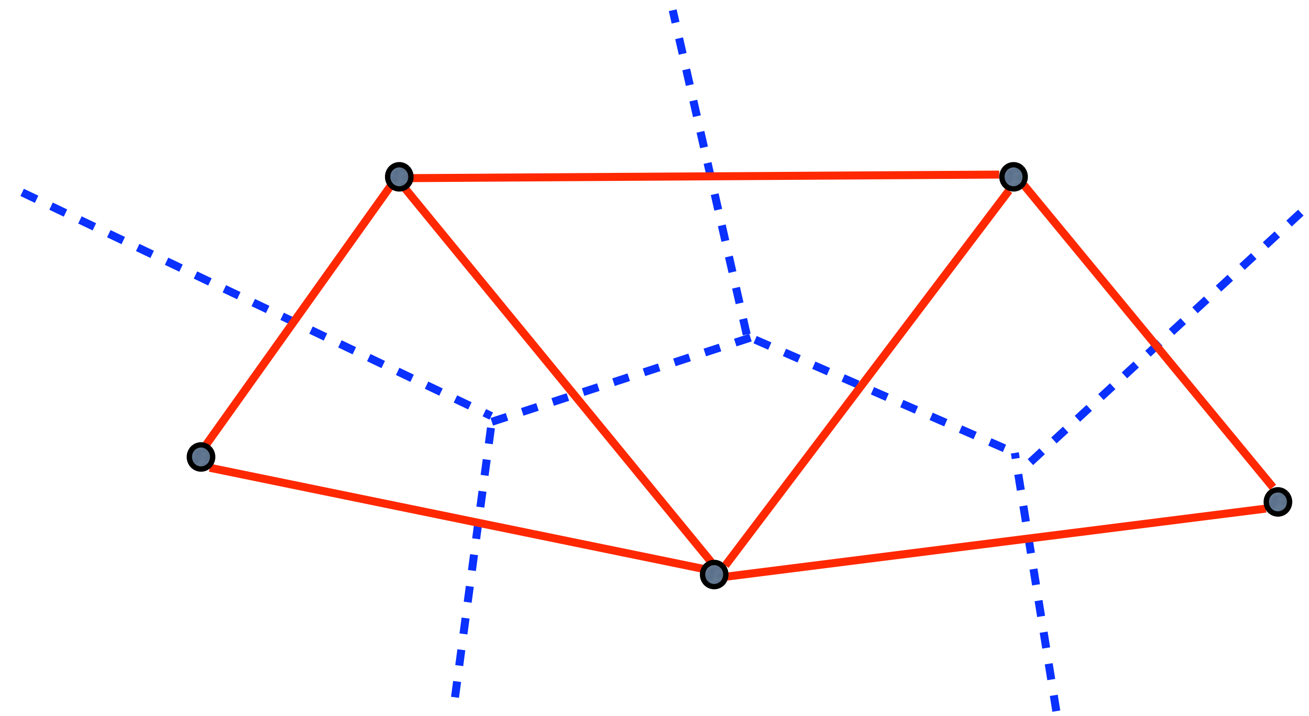


* Delaunay graph: a strategy profile f, there exists an edge (i, j) if i, j are neighbors.

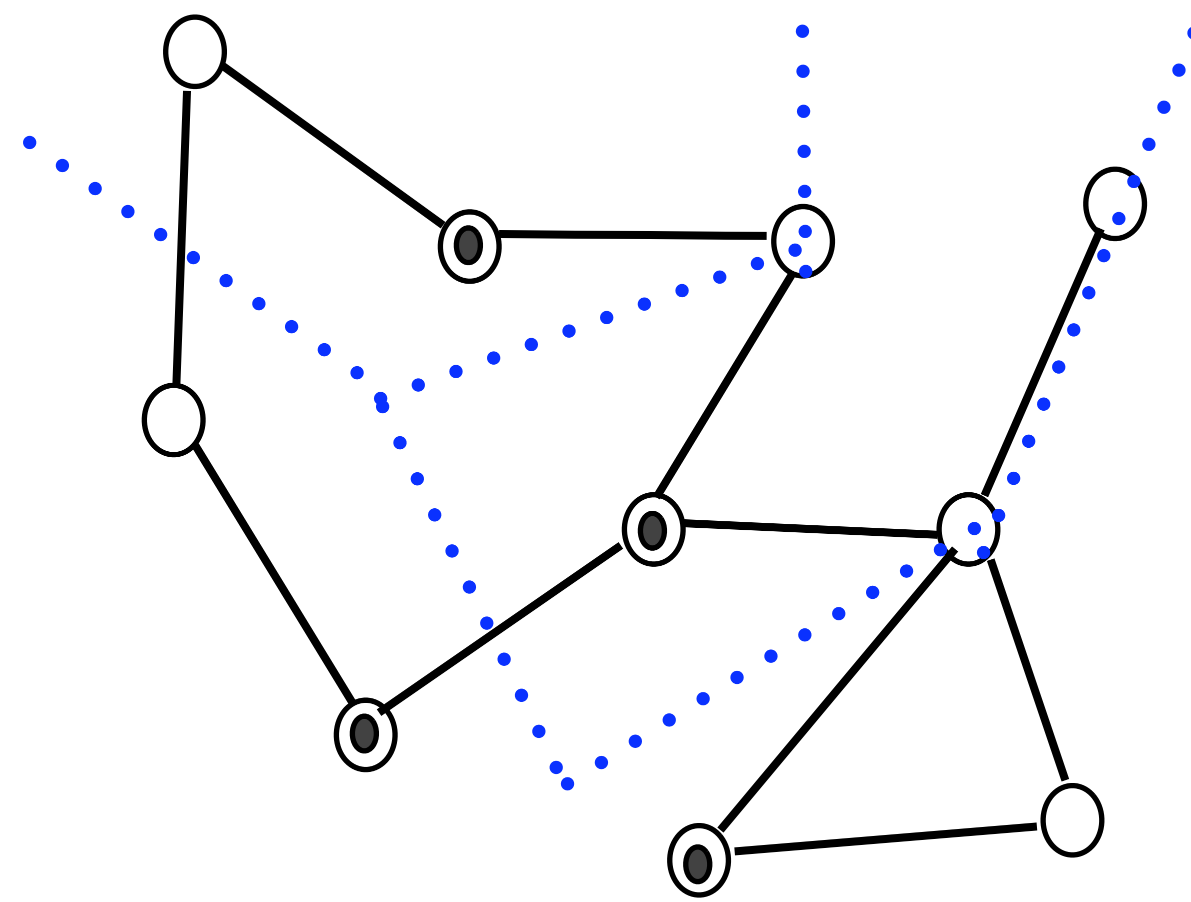


Delaunay triangulation

* Delaunay triangulation:

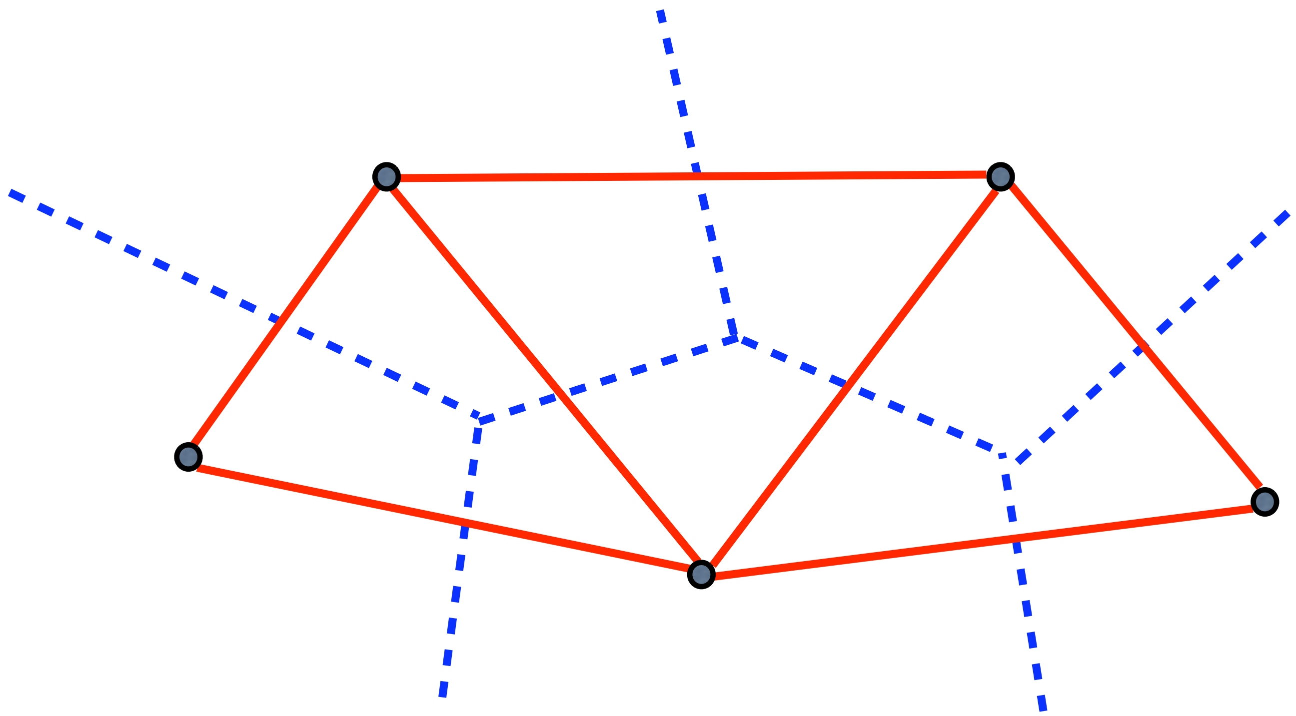


* Delaunay graph: a strategy profile f, there exists an edge (i,j) if i,j are neighbors.

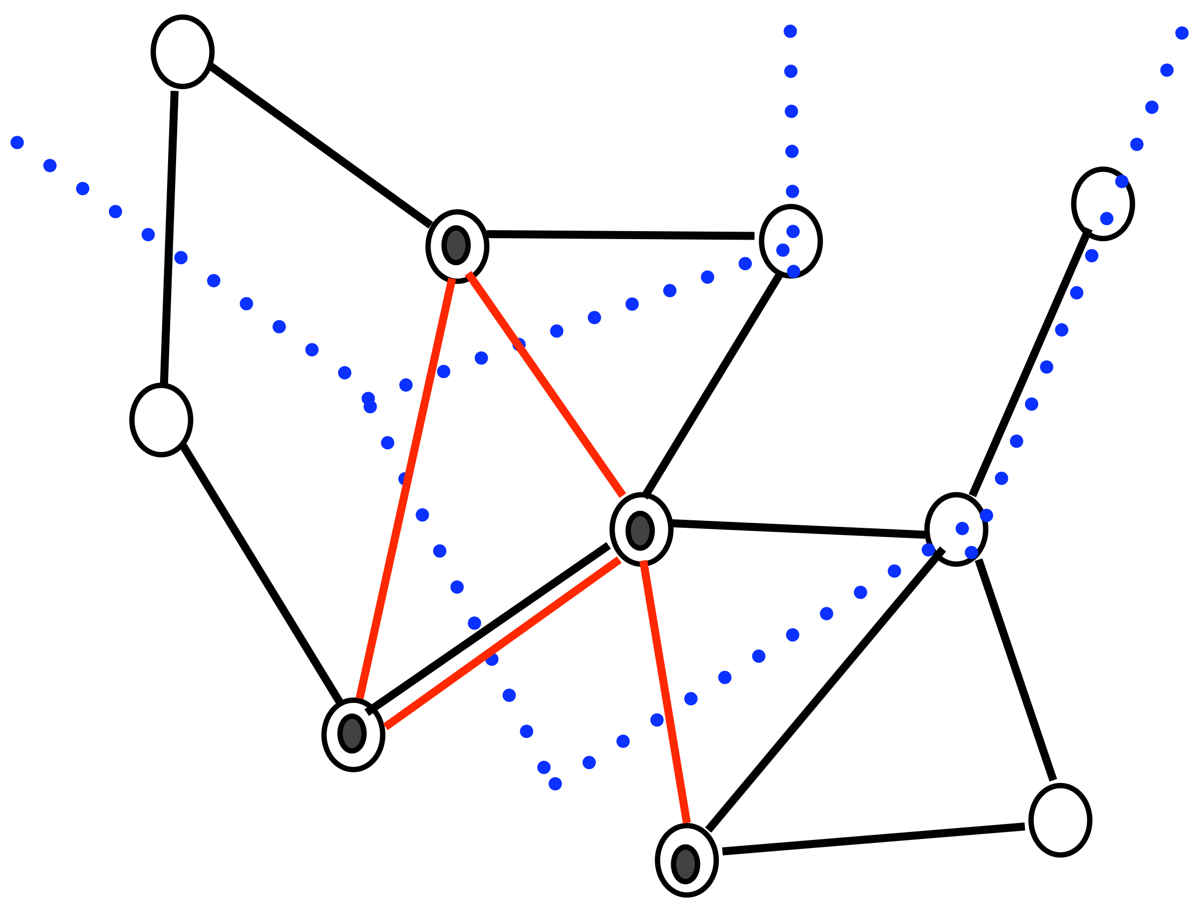


Delaunay triangulation

* Delaunay triangulation:



* Delaunay graph: a strategy profile f, there exists an edge (i, j) if i, j are neighbors.

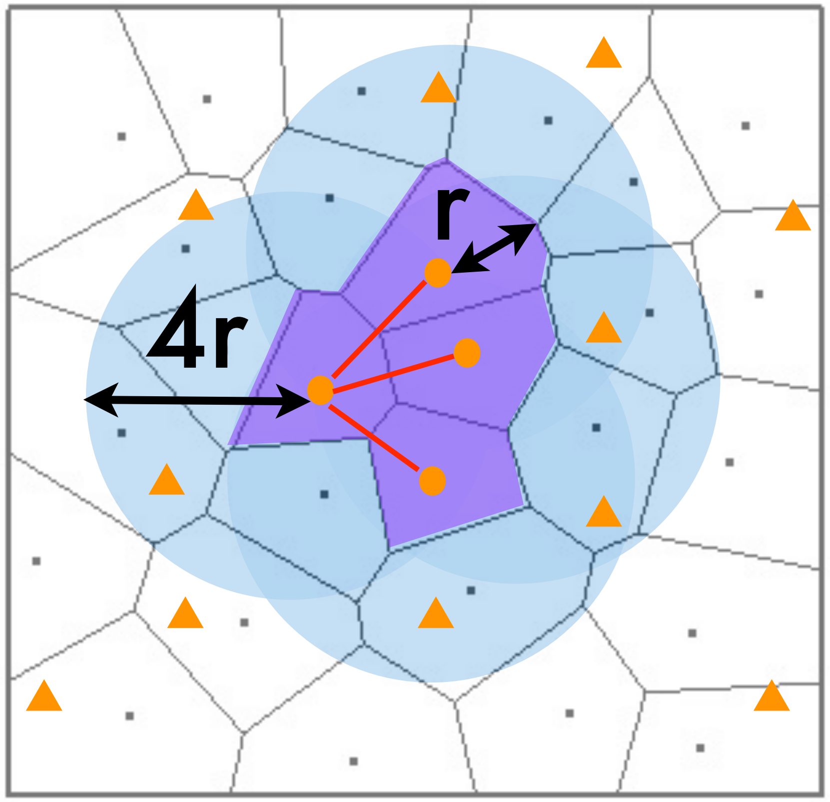


Social cost discrepancy

Theorem: The social cost discrepancy is $\Omega(\sqrt{n/k})$ and $O(\sqrt{kn})$

Proof:

- Consider two equilibria ● and ▲.
- Partition the Delaunay graph corresponding to ● into regions.
- Showing that each location of ▲ is not so far from a region above (compared to the diameter of the region).



Improvements

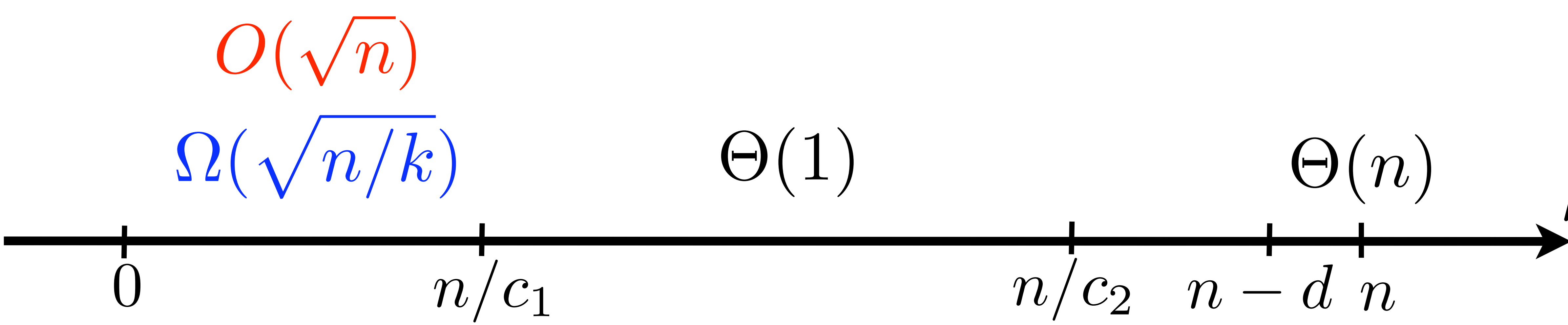
Theorem:

- If $k \leq n/4$ then the discrepancy is $O(\sqrt{n})$
- If there exist constants $c_1 \geq c_2, n_0$ such that:

  $\forall\ n \geq n_0 : n/c_1 \leq k \leq n/c_2$

  then the discrepancy is $\Theta(1)$
- If there exists constant d such that $k \geq n - d$

  then the discrepancy is $\Theta(n)$



Improvements

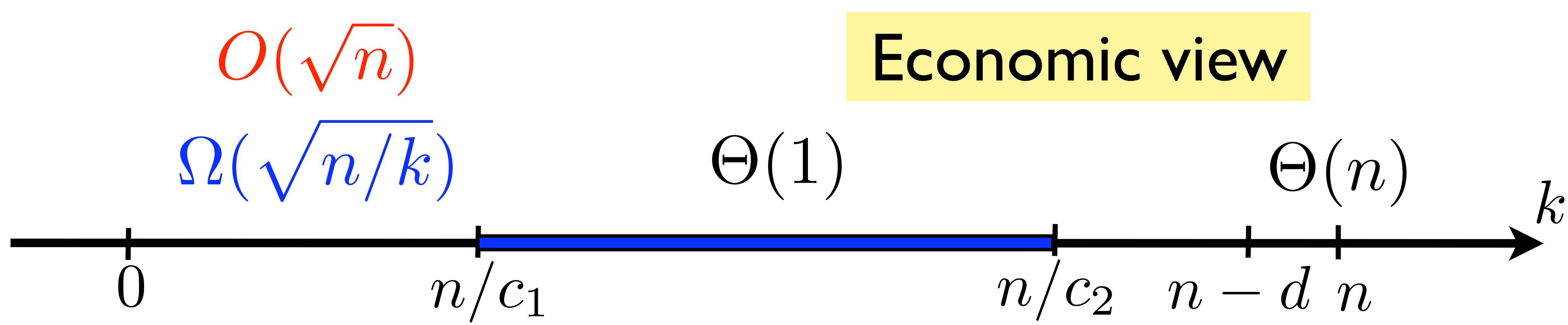
Theorem:

- If $k \leq n/4$ then the discrepancy is $O(\sqrt{n})$
- If there exist constants $c_1 \geq c_2, n_0$ such that:

  $\forall\ n \geq n_0 : n/c_1 \leq k \leq n/c_2$

  then the discrepancy is $\Theta(1)$
- If there exists constant d such that $k \geq n - d$

  then the discrepancy is $\Theta(n)$



Scheduling Games in the Dark

Scheduling Games

- n jobs (players) and m machines: a job chooses a machine to execute. The processing time of job i on machine j is p_{ij}
- The cost c_i of a job i is its completion time.
- The social cost is the makespan, i.e. $\max_i c_i$
- Each machine specifies a policy how jobs assigned to the machine are to be scheduled.

Scheduling Games

* n jobs (players) and m machines: a job chooses a machine to execute. The processing time of job i on machine j is p_{ij}
* The cost c_i of a job i is its completion time.
* The social cost is the makespan, i.e. $\max_i c_i$
* Each machine specifies a policy how jobs assigned to the machine are to be scheduled.
    - Eg: Shortest Processing Time First (SPT)

Scheduling Games

- n jobs (players) and m machines: a job chooses a machine to execute. The processing time of job i on machine j is p_{ij}
- The cost c_i of a job i is its completion time.
- The social cost is the makespan, i.e. $\max_i c_i$
- Each machine specifies a policy how jobs assigned to the machine are to be scheduled.
  - Eg: Shortest Processing Time First (SPT)

Scheduling Games

* n jobs (players) and m machines: a job chooses a machine to execute. The processing time of job i on machine j is p_{ij}
* The cost c_i of a job i is its completion time.
* The social cost is the makespan, i.e. $\max_i c_i$
* Each machine specifies a policy how jobs assigned to the machine are to be scheduled.
    - Eg: Shortest Processing Time First (SPT)

Non-clairvoyant policies

* Typically, a policy depends on the processing time of jobs assigned to the machine.

Non-clairvoyant policies

* Typically, a policy depends on the processing time of jobs assigned to the machine.
* What about policies that do not require this knowledge?
    - Private information of jobs
    - Jobs cannot influence on their completion time by misreporting their processing time
    - Incomplete information games

Non-clairvoyant policies

* Typically, a policy depends on the processing time of jobs assigned to the machine.

* What about policies that do not require this knowledge?
  - Private information of jobs
  - Jobs cannot influence on their completion time by misreporting their processing time
  - Incomplete information games

Non-clairvoyant policies

existence
Nash equilibrium

small PoA

Natural policies

* RANDOM: schedules jobs in a random order.

  In the strategy profile σ, i is assigned to j:

$$c_i = p_{ij} + \frac{1}{2} \sum_{i' : \sigma(i') = j, i' \neq i} p_{i'j}$$

Natural policies

* RANDOM: schedules jobs in a random order.

  In the strategy profile σ, i is assigned to j:

$$c_i = p_{ij} + \frac{1}{2} \sum_{i':\sigma(i')=j, i' \neq i} p_{i'j}$$

* EQUI: schedules jobs in parallel, assigning each job an equal fraction of the processor.

Natural policies

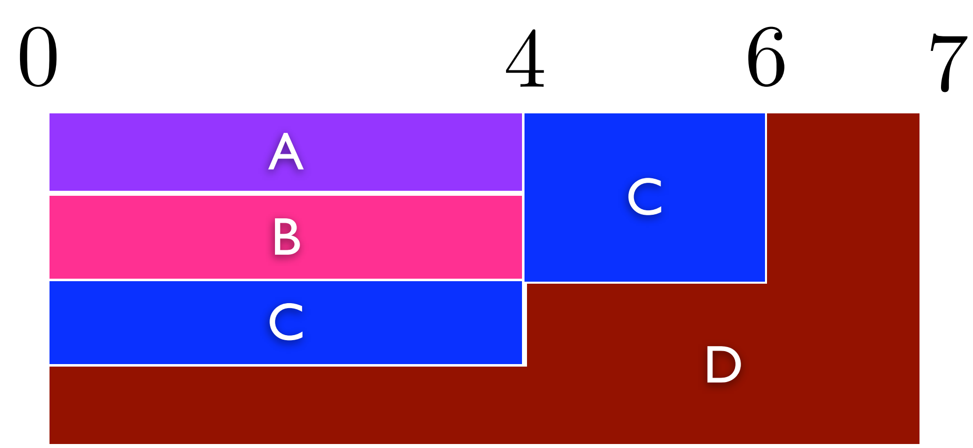
* RANDOM: schedules jobs in a random order.

  In the strategy profile σ, i is assigned to j:

$$c_i = p_{ij} + \frac{1}{2} \sum_{i' : \sigma(i') = j, i' \neq i} p_{i'j}$$

* EQUI: schedules jobs in parallel, assigning each job an equal fraction of the processor.

$$p_A = 1 \quad p_B = 1 \quad p_C = 2 \quad p_D = 3$$



Natural policies

- RANDOM: schedules jobs in a random order.

  In the strategy profile σ, i is assigned to j:

$$c_i = p_{ij} + \frac{1}{2} \sum_{i':\sigma(i')=j, i'\neq i} p_{i'j}$$

- EQUI: schedules jobs in parallel, assigning each job an equal fraction of the processor.

  If there are k jobs on machine j s.t: $p_{1j} \leq \ldots \leq p_{kj}$

$$c_i = p_{1j} + \ldots + p_{i-1,j} + (k - i + 1) p_{ij}$$

Definitions

* Def: A job i is balanced if $\max p_{ij} / \min p_{ij} \leq 2$

* Def of models:
  - Identical machines: $p_{ij} = p_i \; \forall j$ for some length p_i
  - Uniform machines: $p_{ij} = p_i / s_j$ for some speed s_j
  - Unrelated machines: p_{ij} arbitrary

Definitions

* Def: A job is unhappy if it can decrease its cost by changing the strategy (other players' strategies are fixed)

* Def: Best-response dynamic is a process that let an arbitrary unhappy player (job) make a best response -- a strategy that maximizes player's utility.

Our results

* Existence of equilibria: potential argument.

	identical	*uniform*	*unrelated*
RANDOM	*NE*		*non-convergence*
EQUI	*NE*	*NE*	*NE*

Idea:
- Best-response dynamic may cycle
- New dynamic to break the cycle.

Our results

* Existence of equilibria: potential argument.

	identical	*uniform*	*unrelated*
RANDOM	*NE*	*NE for balanced jobs*	*non-convergence*
EQUI	*NE*	*NE*	*NE*

Idea:

- Best-response dynamic may cycle
- New dynamic to break the cycle.

Our results

* Existence of equilibria: potential argument.

	identical	*uniform*	*unrelated*
RANDOM	*NE*	*NE for balanced jobs*	*non-convergence*
EQUI	*NE*	*NE*	*NE*

Idea:
- Best-response dynamic may cycle
- New dynamic to break the cycle.

RANDOM, uniform machines

RANDOM, uniform machines

- Jobs have length $p_1 \leq p_2 \leq \ldots \leq p_n$
- Machines have speed $s_1 \geq s_2 \geq \ldots \geq s_m$

$$p_{ij} = p_i / s_j$$

RANDOM, uniform machines

* Jobs have length $p_1 \leq p_2 \leq \ldots \leq p_n$
* Machines have speed $s_1 \geq s_2 \geq \ldots \geq s_m$

$$p_{ij} = p_i / s_j$$

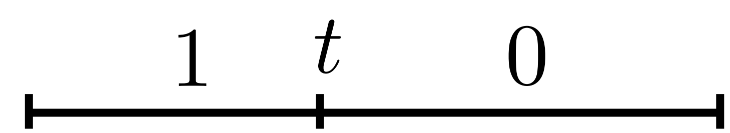
* Dynamic: among all unhappy jobs, let the one with the greatest index make a best move.

Potential function

Potential function

* For any strategy profile σ, let t be the unhappy job with greatest index.

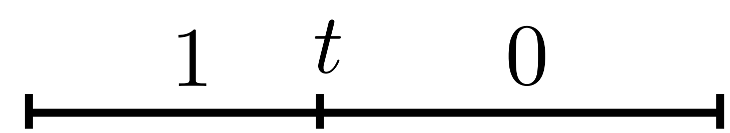
$$f_\sigma(i) = \begin{cases} 1 & \text{if } 1 \leq i \leq t, \\ 0 & \text{otherwise.} \end{cases}$$



Potential function

* For any strategy profile σ, let t be the unhappy job with greatest index.

$$f_\sigma(i) = \begin{cases} 1 & \text{if } 1 \leq i \leq t, \\ 0 & \text{otherwise.} \end{cases}$$

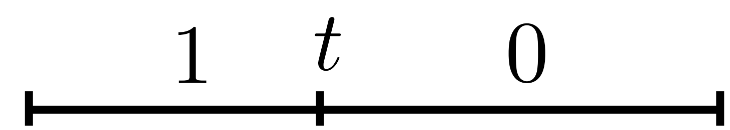


* $\Phi(\sigma) = (f_\sigma(1), s_{\sigma(1)}, \ldots, f_\sigma(n), s_{\sigma(n)})$ lex. decreases

Potential function

* For any strategy profile σ, let t be the unhappy job with greatest index.

$$f_\sigma(i) = \begin{cases} 1 & \text{if } 1 \le i \le t, \\ 0 & \text{otherwise.} \end{cases}$$



* $\Phi(\sigma) = (f_\sigma(1), s_{\sigma(1)}, \dots, f_\sigma(n), s_{\sigma(n)})$ lex. decreases

* Dominance: either the number of unhappy players decreases or the lexicographical order of machines' speeds are decreased.

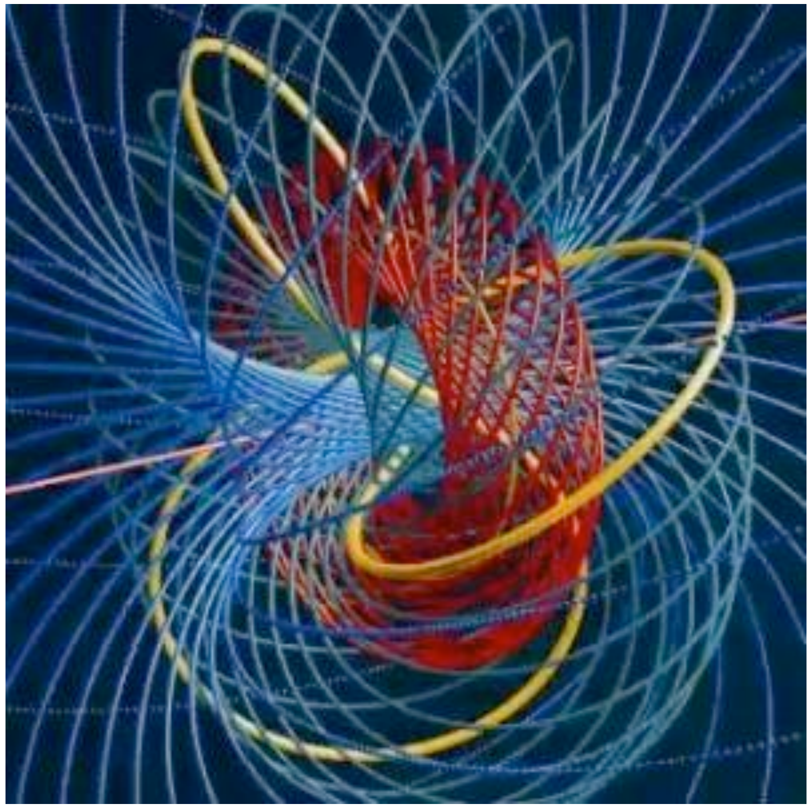
Our results

* Existence of equilibria: potential argument.

	identical	*uniform*	*unrelated*
RANDOM	*NE*	*NE for balanced jobs*	*non-convergence*
EQUI	*NE*	*NE*	*NE*

Idea:

- Best-response dynamic may cycle
- New dynamic to break the cycle.
- Dominance: either the number of unhappy players decreases or the lexicographical order of machines' speeds are decreased.

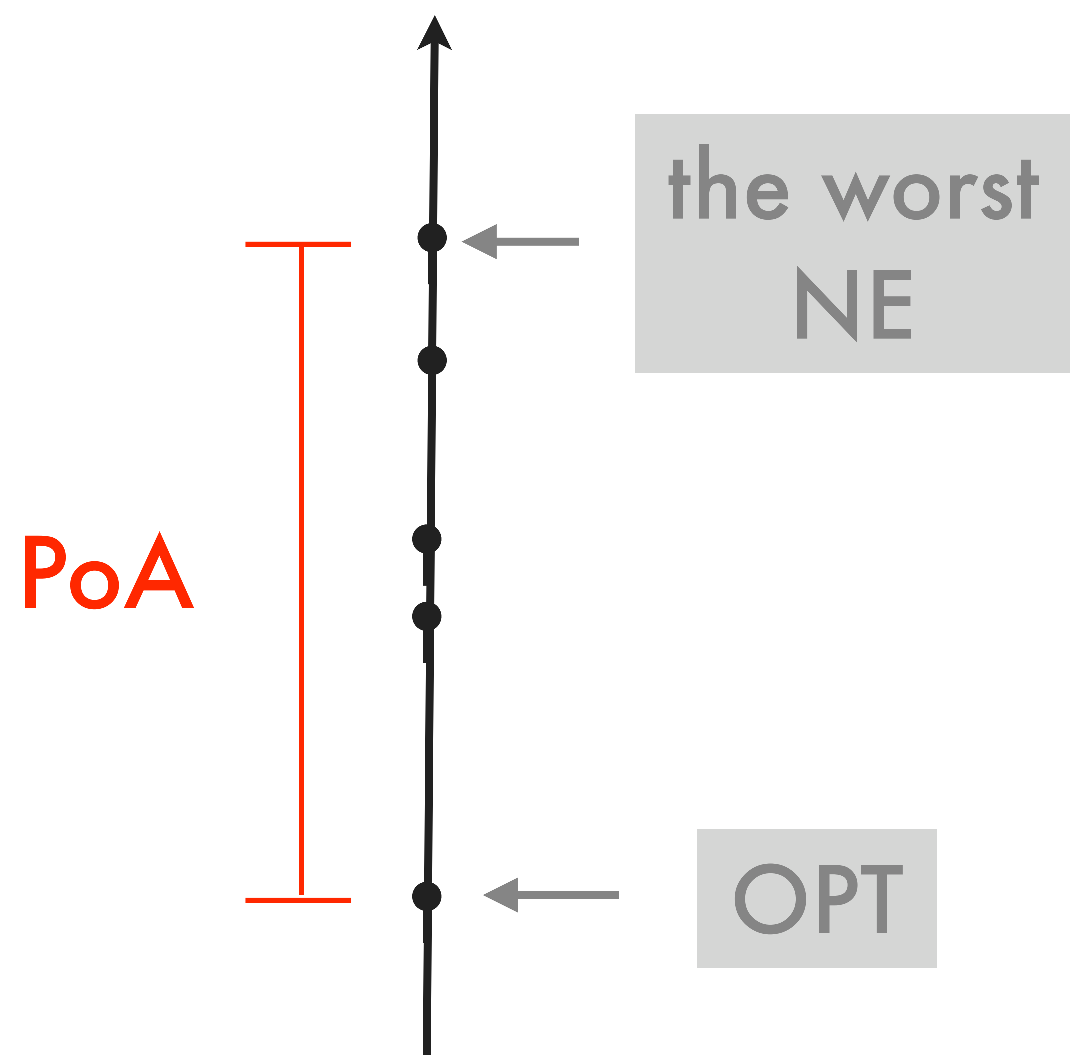


Our results

- *Theorem:* For unrelated machines, the PoA of policy EQUI is at most 2m – interestingly, that matches the best clairvoyant policy.

- PoA is not increased when processing times are unknown to the machines.

- The knowledge about jobs' characteristics is not necessarily needed.



Online Mechanism Design

Mechanism Design

Define the game

Goal: self-interested behavior yields desired outcomes.

Online Auction

* A company produces one perishable item per time unit (items have to be immediately delivered to bidders, e.g. electricity, ice-cream, ...)

Online Auction

* A company produces one perishable item per time unit (items have to be immediately delivered to bidders, e.g. electricity, ice-cream, ...)
* Single-minded bidders arrive online: a customer arrives at r_i, pays w_i if he receives k_i items before deadline d_i, otherwise he pays nothing.

Online Auction

* A company produces one perishable item per time unit (items have to be immediately delivered to bidders, e.g. electricity, ice-cream, ...)
* Single-minded bidders arrive online: a customer arrives at r_i, pays w_i if he receives k_i items before deadline d_i, otherwise he pays nothing.
* Opt. prob: maximize the welfare $\sum_i w_i$ over all satisfied bidders.

Online Auction

* A company produces one perishable item per time unit (items have to be immediately delivered to bidders, e.g. electricity, ice-cream, ...)

* Single-minded bidders arrive online: a customer arrives at r_i, pays w_i if he receives k_i items before deadline d_i, otherwise he pays nothing.

* Opt. prob: maximize the welfare $\sum_i w_i$ over all satisfied bidders.

* Mechanism design:
  - w_i are private
  - Bidders may misreport their value. They bid b_i

Truthful Auction Design

Truthful Auction Design

Auction:
receives all bids

allocation algorithm:
determine the set of satisfied bidders

payment algorithm:
determine how much a bidder has to pay

Truthful Auction Design

Auction:
receives all bids

allocation algorithm:
determine the set of satisfied bidders

payment algorithm:
determine how much a bidder has to pay

$$u_i = \begin{cases} w_i - p_i & \text{if satisfied,} \\ 0 & \text{otherwise.} \end{cases}$$

Goal: self-interested behavior yields **truthfulness**, $b_i = w_i$

Truthful Auction Design

Auction:
receives all bids

allocation algorithm:
determine the set of satisfied bidders

payment algorithm:
determine how much a bidder has to pay

Truthful Auction Design

Auction:
receives all bids

allocation algorithm:
determine the set of satisfied bidders

payment algorithm:
determine how much a bidder has to pay

monotone:
a winner still wins if he raises his bid

critical bid:
the smallest bid that a winner needs to bid in order to win.

Monotone algorithm

- Our problem:
  - design a monotone allocation algorithm
  - verify whether the critical payment scheme can be computed efficiently.

Monotone algorithm

- Our problem:
  - design a monotone allocation algorithm
  - verify whether the critical payment scheme can be computed efficiently.

- Maximizing the welfare $\sum_i w_i$ is NP-hard even offline.

Monotone algorithm

- Our problem:
  - design a monotone allocation algorithm
  - verify whether the critical payment scheme can be computed efficiently.

- Maximizing the welfare $\sum_i w_i$ is NP-hard even offline.

- Scheduling problem: $1|r_i - online,\ pmtn|\sum_i w_i$

  ... with monotone algorithm

Online Algorithm

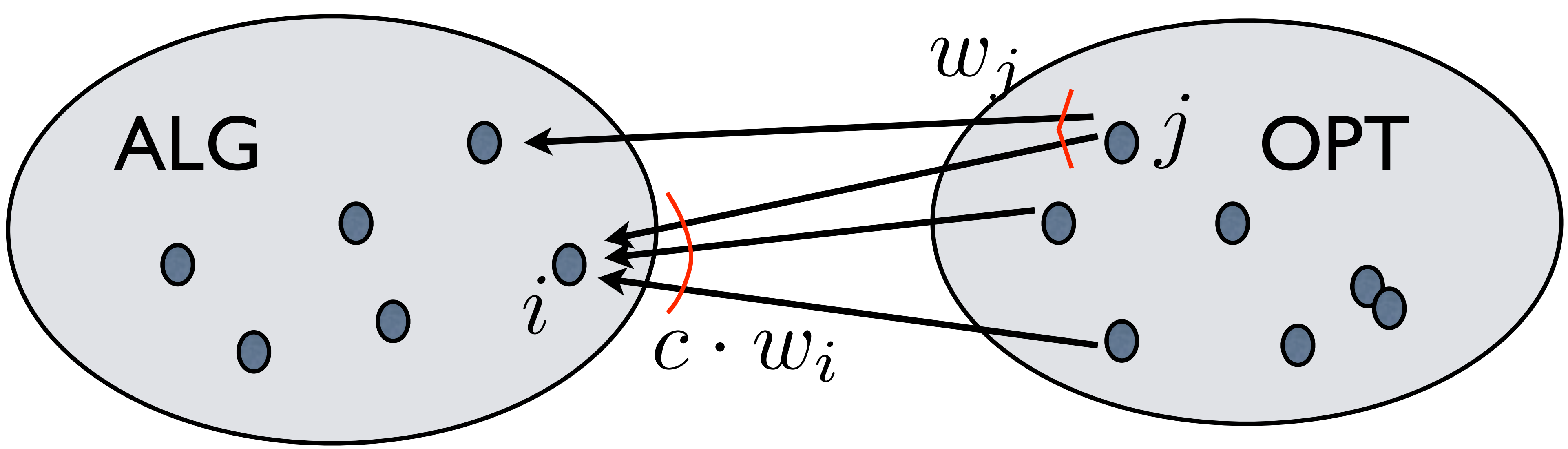
* Def: an online algorithm ALG is c-competitive if for any instance I, the outcome $c \cdot ALG(I) \geq OPT(I)$

* Technique: charging scheme

Online Algorithm

* Def: an online algorithm ALG is c-competitive if for any instance I, the outcome $c \cdot ALG(I) \geq OPT(I)$

* Technique: charging scheme



Our results

- *Theorem:* If $k_i \leq k \; \forall i$ then
  - The Smith algorithm which serves the bidder that maximizes b_i/q_i is $2k$-competitive where q_i is the remaining demands of bidder i .
  - The algo which serves the bidder that maximizes $b_i \cdot \alpha^{q_i - 1}$ is $\Theta(k/\log k)$-competitive where $\alpha = 1 - (1-\epsilon)^2 \cdot (\ln k)/k$.
  - There exists a 5-competitive alg. if $k_i = k \; \forall i$

Our results

* *Theorem:* If $k_i \leq k \ \forall i$ then
  - The Smith algorithm which serves the bidder that maximizes b_i/q_i is $2k$-competitive where q_i is the remaining demands of bidder i .
  - The algo which serves the bidder that maximizes $b_i \cdot \alpha^{q_i - 1}$ is $\Theta(k/\log k)$-competitive where $\alpha = 1 - (1-\epsilon)^2 \cdot (\ln k)/k$. Optimal alg.
  - There exists a 5-competitive alg. if $k_i = k \ \forall i$

Our results

* *Theorem:* If $k_i \leq k \ \forall i$ then
  - The Smith algorithm which serves the bidder that maximizes b_i/q_i is $2k$-competitive where q_i is the remaining demands of bidder i .
  - The algo which serves the bidder that maximizes $b_i \cdot \alpha^{q_i - 1}$ is $\Theta(k/\log k)$-competitive where $\alpha = 1 - (1-\epsilon)^2 \cdot (\ln k)/k$. Optimal alg.
  - There exists a 5-competitive alg. if $k_i = k \ \forall i$
* *Proof:* Using general charging scheme.

Our results

* *Theorem:* If $k_i \leq k \ \forall i$ then

  - The Smith algorithm which serves the bidder that maximizes b_i/q_i is $2k$-competitive where q_i is the remaining demands of bidder i .

  - The algo which serves the bidder that maximizes $b_i \cdot \alpha^{q_i - 1}$ is $\Theta(k/\log k)$-competitive where $\alpha = 1 - (1-\epsilon)^2 \cdot (\ln k)/k$. Optimal alg.

  - There exists a 5-competitive alg. if $k_i = k \ \forall i$

* *Corollary:* there exists truthful optimal mechanism with the same competitive ratio.

Summary

Given game

Players ⟷ Φ ⟷ Social objective

(maximize their utilities)

Summary

Given game

Players
(maximize their utilities) $\longleftrightarrow$ Φ $\longleftrightarrow$ Social objective

X ?

Summary

Given game

Players
(maximize their utilities)

Φ

Social objective

X ?

Design games (algorithms)
(a little more control,
much better quality of outcomes)

Thank you!