Algorithmic Game Theory

Nguyen Kim Thang

LIAFA, 18/2/09

Game Theory + Algorithms

Game Theory + Algorithms

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

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).

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

Outline

* Existence and inefficiency of pure Nash equilibrium
    * Scheduling Games in the Dark
* Online Algorithmic Mechanism Design
    * Online Auction with single-minded customers

Nash Equilibrium

Nash Equilibrium

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

Nash Equilibrium

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

Mixed equilibrium

Pure equilibrium

Nash Equilibrium

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

Mixed equilibrium

choose a distribution over strategies

Pure equilibrium

Nash Equilibrium

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

Mixed equilibrium

choose a distribution over strategies

Pure equilibrium

deterministically choose a strategy

Nash Equilibrium

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

Mixed equilibrium

choose a distribution over strategies

always exists (by Nash)

Pure equilibrium

deterministically choose a strategy

Nash Equilibrium

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

Mixed equilibrium

choose a distribution over strategies

always exists (by Nash)

Pure equilibrium

deterministically choose a strategy

does not necessarily exist

Nash Equilibrium

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

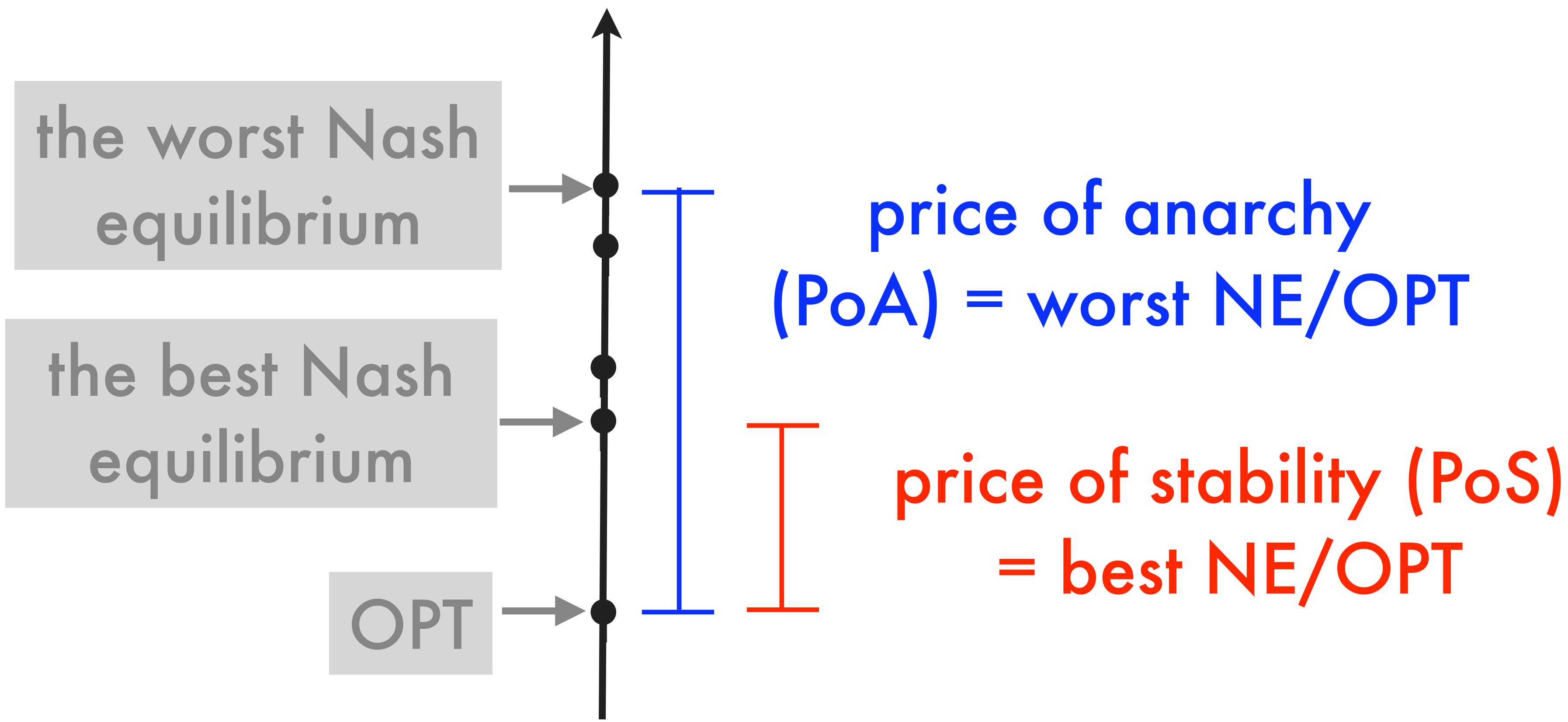
Mixed equilibrium	Pure equilibrium
choose a distribution over strategies	deterministically choose a strategy
always exists (by Nash)	does not necessarily exist

* Potential games: admit a function such that if a player change her strategy to get a better utility then the function strictly decreases.

Inefficiency of equilibria

social objective function

Good equilibria ?



Scheduling Game

Scheduling Game

* 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}

Scheduling Game

* 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.

Scheduling Game

- 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. $\max_i c_i$

Scheduling Game

* 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 Game

- 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 Game

- 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?
  - Incomplete information games
  - Private information of jobs
  - Jobs cannot influence on their completion time by misreporting their processing time

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?
  - Incomplete information games
  - Private information of jobs
  - Jobs cannot influence on their completion time by misreporting their processing time

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:

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

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

Natural policies

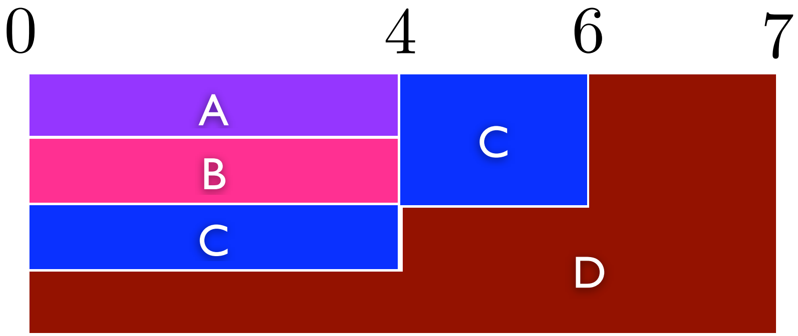
* RANDOM: schedules jobs in a random order.

  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: $p_{1j} \leq \ldots \leq p_{kj}$

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

Models

* 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

Existence of equilibrium

- Theorem:
  - The game under EQUI policy is a potential game.
  - The game under RANDOM policy is a potential game for 2 unrelated machines but it is not for more than 3 machines. For uniform machines, balanced jobs, there always exists equilibrium.

RANDOM, uniform machines

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

RANDOM, uniform machines

* Def: A job is unhappy if it can decrease its cost by changing the strategy (other players' strategies are fixed)
* Def: a best move of a job is the strategy which minimizes the cost of the job (while other players' strategies are fixed)

RANDOM, uniform machines

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

* Def: a best move of a job is the strategy which minimizes the cost of the job (while other players' strategies are fixed)

* 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

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

* Def: a best move of a job is the strategy which minimizes the cost of the job (while other players' strategies are fixed)

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

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

* Lemma: Consider a job i making a best move from a to b. If there is a new unhappy job with index greater than i, then $s_a > s_b$

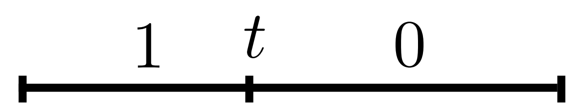
Potential function

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

Potential function

- Dynamic: among all unhappy jobs, let the one with the greatest index make a best move.
- For any strategy profile σ, let t be the unhappy job with greatest index.

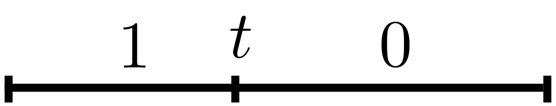
$$f_\sigma(i) = \begin{cases} 1 & \text{if } 1 \le i \le t, \\ 0 & \text{otherwise.} \end{cases}$$



Potential function

- **Dynamic:** among all unhappy jobs, let the one with the greatest index make a best move.
- For any strategy profile σ, let t be the unhappy job with greatest index.

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



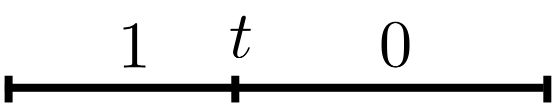
- $\Phi(\sigma) = (f_\sigma(1), s_{\sigma(1)}, \ldots, f_\sigma(n), s_{\sigma(n)})$ lex. decreases

Potential function

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

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

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



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

  If $t' < t$

$$\Phi(\sigma) = (1, s_{\sigma(1)}, \ldots, 1, s_{\sigma(t')}, 1, s_{\sigma(t'+1)}, \ldots, 1, s_{\sigma(t)}, \ldots)$$

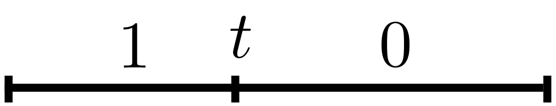
$$\Phi(\sigma') = (1, s_{\sigma(1)}, \ldots, 1, s_{\sigma(t')}, 0, s_{\sigma(t'+1)}, \ldots, 0, s_{\sigma'(t)}, \ldots)$$

Potential function

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

* 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

  If $t' > t$

$$\Phi(\sigma) = (1, s_{\sigma(1)}, \ldots, 1, s_{\sigma(t)}, \ldots, 0, s_{\sigma(t')}, \ldots)$$

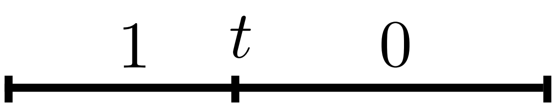
$$\Phi(\sigma') = (1, s_{\sigma(1)}, \ldots, 1, s_{\sigma'(t)}, \ldots, 0, s_{\sigma(t')}, \ldots)$$

Potential function

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

* 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

  If $t' > t$

$$\Phi(\sigma) = (1, s_{\sigma(1)}, \dots, 1, s_{\sigma(t)}, \dots, 0, s_{\sigma(t')}, \dots)$$

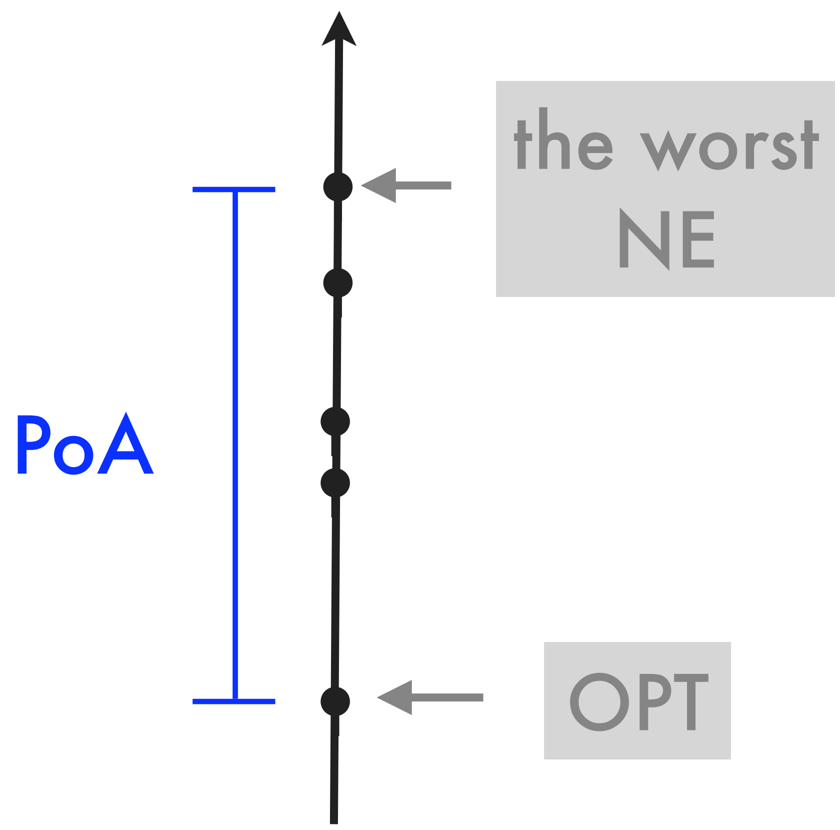
$$\Phi(\sigma') = (1, s_{\sigma(1)}, \dots, 1, s_{\sigma'(t)}, \dots, 0, s_{\sigma(t')}, \dots)$$

by Lemma: $s_{\sigma(t)} > s_{\sigma'(t)}$

Inefficiency

* 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.



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 customers, e.g. electricity, ice-cream, ...)
* Single-minded customers 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 customers.

Online Auction

* A company produces one perishable item per time unit (items have to be immediately delivered to customers, e.g. electricity, ice-cream, ...)
* Single-minded customers 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 customers.
* Mechanism design:
  - w_i are private
  - Customers may misreport their value. They bid b_i

Mechanism Design (MD)

Mechanism:
receives all bids

Mechanism Design (MD)

Mechanism:
receives all bids

allocation algorithm:
determine the set of
satisfied customers

Mechanism Design (MD)

Mechanism:
receives all bids

allocation algorithm:
determine the set of satisfied customers

payment algorithm:
determine how much a customer has to pay

Mechanism Design (MD)

Mechanism:
receives all bids

allocation algorithm:
determine the set of
satisfied customers

payment algorithm:
determine how much
a customer has to pay

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

Mechanism Design (MD)

Mechanism:
receives all bids

allocation algorithm:
determine the set of satisfied customers

payment algorithm:
determine how much a customer 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**, $b_i = w_i$

Mechanism Design (MD)

Mechanism:
receives all bids

allocation algorithm:
determine the set of satisfied customers

payment algorithm:
determine how much a customer has to pay

Mechanism Design (MD)

Mechanism:
receives all bids

allocation algorithm:
determine the set of satisfied customers

payment algorithm:
determine how much a customer has to pay

monotone:
a winner still win if he raises his bid

Mechanism Design (MD)

Mechanism:
receives all bids

allocation algorithm:
determine the set of satisfied customers

payment algorithm:
determine how much a customer has to pay

monotone:
a winner still win if he raises his bid

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

Truthful MD

- Theorem: for single-parameter domain, a mechanism is truthful iff its allocation algo is monotone and it uses the critical payment scheme.

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

Online Algorithm

* Maximizing the welfare $\sum_i w_i$ is hard.

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

Online Algorithm

- Maximizing the welfare $\sum_i w_i$ is hard.

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

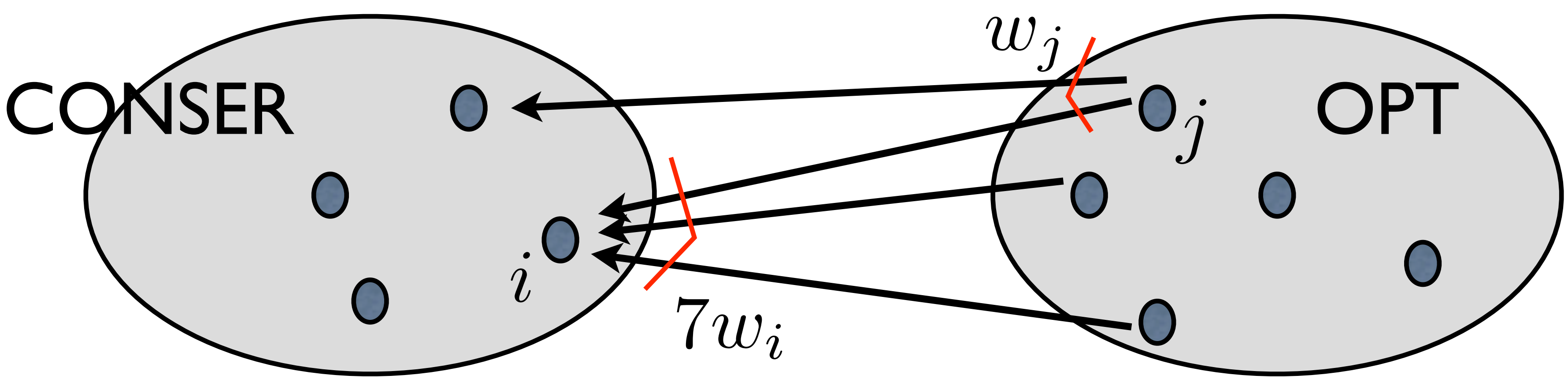
- Theorem: if all $k_i = k$ then there exists a 7-competitive truthful mechanism.

Algorithm

- The CONSERVATIVE algo:
    - if there is no currently running job, serve the pending one with highest value
    - still schedule the current customer except there is a new one with value at least 2 that of the current customer

Algorithm

- The CONSERVATIVE algo:
  - if there is no currently running job, serve the pending one with highest value
  - still schedule the current customer except there is a new one with value at least 2 that of the current customer
- Proof:
  - the algorithm is monotone
  - 7-competitive by a charging scheme



Proof (sketch)

CONSER

OPT

Proof (sketch)

$$\begin{array}{llll}
\text{CONSER} & & & \\
 & ? & ? & ? \\
 & \Big\uparrow w_j/k & \Big\uparrow w_j/k & \Big\uparrow w_j/k \\
\text{OPT} & \boxed{(j,k)} & \boxed{(j,b)} & \boxed{(j,1)}
\end{array}$$

Proof (sketch)

CONSER

OPT $\boxed{(j,k)}$ $\boxed{(j,b)}$ $\boxed{(j,1)}$

Proof (sketch)

CONSER $\boxed{(j,1)}$

OPT $\boxed{(j,k)}$ $\boxed{(j,b)}$ $\boxed{(j,1)}$

- type 1: if j is completed by CONSER

Proof (sketch)

CONSER

OPT $\boxed{(j,k)}$ $\boxed{(j,b)}$ $\boxed{(j,1)}$

- type 1: if j is completed by CONSER

Proof (sketch)

- type 1: if j is completed by CONSER
- type 2: if $2w_i > w_j$

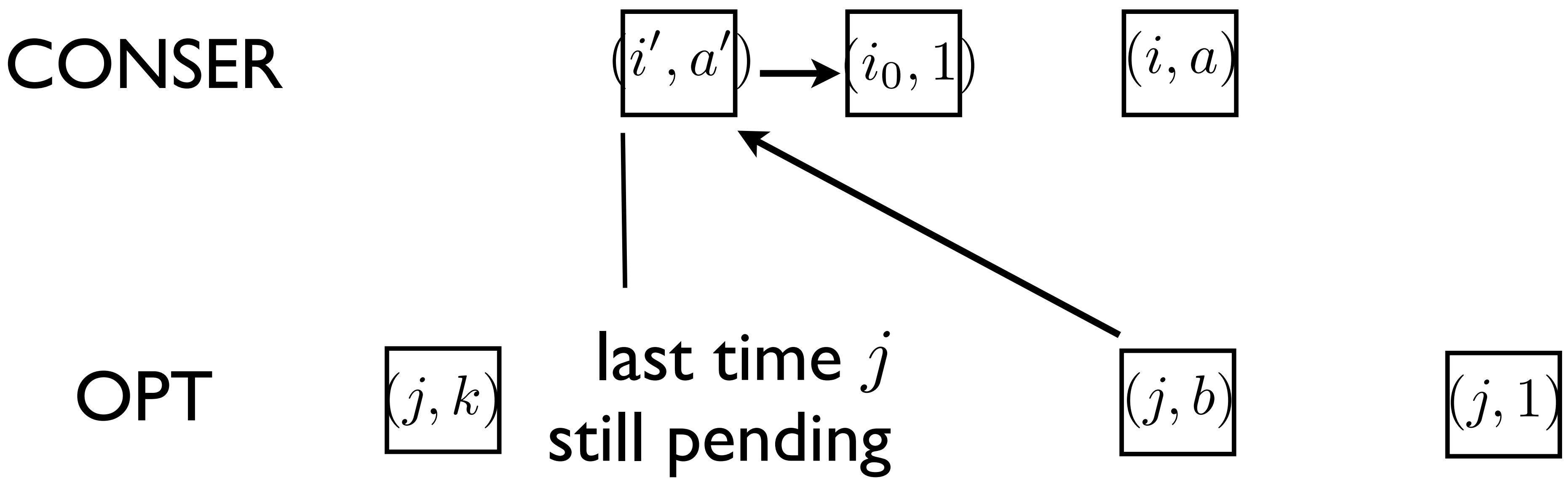
Proof (sketch)

CONSER

OPT $\boxed{(j,k)}$ $\boxed{(j,b)}$ $\boxed{(j,1)}$

- type 1: if j is completed by CONSER
- type 2: if $2w_i > w_j$

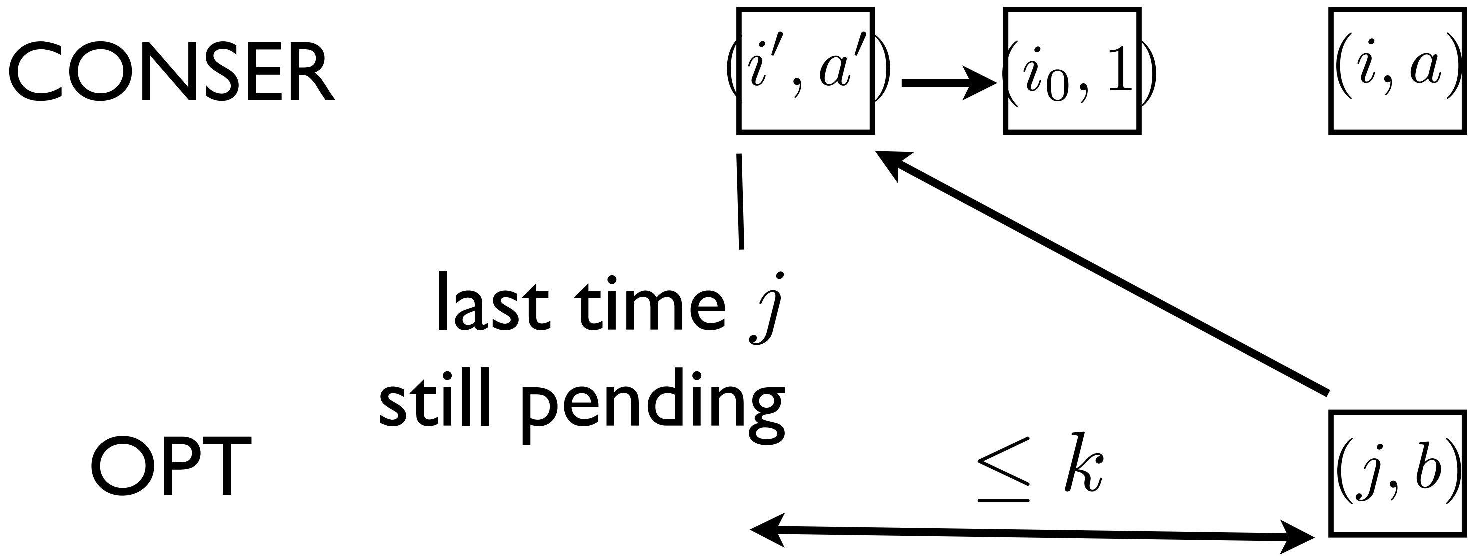
Proof (sketch)



- type 1: if j is completed by CONSER
- type 2: if $2w_i > w_j$
- type 3: otherwise $2w_i \le w_j$, j is not pending.

  $2w_{i'} > w_j$ then $2w_{i_0}/k > w_j/k$

Proof (sketch)



- Observation: $(i_0, 1)$ receives at most k charges of type 3.

- Summing up all charges, we get 7-competitive.

General case

* Theorem: if all $k_i \leq k$ then there exists a $O(k/\log k)$-competitive truthful mechanism. This mechanism is optimal.

* Proof: more elaborated but the idea is similar.

Conclusion

- ☑ Motivation through two problems.
  - theoretically beautiful
  - real problems, practical importance.
- ☑ Inspired by Game Theory, using technique of Computer Science
- ☐ Inspired by Computer Science, using technique of Game Theory