Scheduling Games in the Dark

Nguyen Kim Thang
(joint work with Christoph Durr)

Aarhus
March 9th, 09

Outline

* Scheduling Games
    - Definition & Motivation
    - Summary of results
* Existence of pure Nash equilibrium
    - Potential argument
* Inefficiency of equilibria
    - Price of anarchy

Scheduling Games

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}

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}

* Such a job-machine assignment σ is a strategy profile. The load of machine j is $\ell_j = \sum_{i:\sigma(i)=j} p_{ij}$

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}

* Such a job-machine assignment σ is a strategy profile. The load of machine j is $\ell_j = \sum_{i:\sigma(i)=j} p_{ij}$

* Each machine specifies a policy how jobs assigned to the machine are to be scheduled (e.g., SPT, LPT, ...).

* The cost c_i of a job i is its completion time.

* The social cost is the makespan, i.e. $\max_j \ell_j = \max_i c_i$

Coordination Mechanism

Coordination Mechanism

* Coordination mechanism aim to design policies such that the self-interested behaviors lead to equilibria with small PoA.

Coordination Mechanism

* Coordination mechanism aim to design policies such that the self-interested behaviors lead to equilibria with small PoA.
* Communication is hard or costly (large-scale autonomous systems in the Internet).

Coordination Mechanism

- Coordination mechanism aim to design policies such that the self-interested behaviors lead to equilibria with small PoA.
- Communication is hard or costly (large-scale autonomous systems in the Internet).
- Policies are designed based on local information.

Coordination Mechanism

- Coordination mechanism aim to design policies such that the self-interested behaviors lead to equilibria with small PoA.
- Communication is hard or costly (large-scale autonomous systems in the Internet).
- Policies are designed based on local information.
  - Strongly local policy: a machine looks only at proc. time of jobs assigned to the machine.

$$\sigma(i) = j \longrightarrow p_{ij}$$

Coordination Mechanism

- Coordination mechanism aim to design policies such that the self-interested behaviors lead to equilibria with small PoA.
- Communication is hard or costly (large-scale autonomous systems in the Internet).
- Policies are designed based on local information.
  - Strongly local policy: a machine looks only at proc. time of jobs assigned to the machine.

    $$\sigma(i) = j \longrightarrow p_{ij}$$

  - Local policy: depends only on the parameters of jobs assigned to it. $\sigma(i) = j \longrightarrow p_{ij'} \; \forall j'$

Policies

* Strongly local policies:

Policies

* Strongly local policies:
  * Shortest Processing Time First (SPT)

Policies

* Strongly local policies:
  - Shortest Processing Time First (SPT)

Policies

- Strongly local policies:
  - Shortest Processing Time First (SPT)

Policies

- Strongly local policies:
  - Shortest Processing Time First (SPT)

  - Longest Processing Time First (LPT)

Policies

- Strongly local policies:
  - Shortest Processing Time First (SPT)

  - Longest Processing Time First (LPT)
  - MAKESPAN: if $\sigma(i) = j$

$$c_i = \ell_j$$

Policies

Policies

- Local policies:
  - Inefficiency-based policy: greedily schedule jobs in increasing order of $\rho_i = p_{ij}/q_i$ where $q_i = \min_{j'} p_{ij'}$

Policies

- Local policies:
  - Inefficiency-based policy: greedily schedule jobs in increasing order of $\rho_i = p_{ij}/q_i$ where $q_i = \min_{j'} p_{ij'}$
- Typically, a policy depends on the processing time of jobs assigned to the machine.

Non-clairvoyant policies

Non-clairvoyant policies

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

Non-clairvoyant policies

* What about policies that do not require this knowledge?
  - Incomplete information games
  - Private information of jobs
  - Jobs can 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.

Natural policies

- RANDOM: schedules jobs in a random order.
  - In the strategy profile σ, i is assigned to j

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

Natural policies

* RANDOM: schedules jobs in a random order.
  - In the strategy profile σ, i is assigned to j

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

  - A job i on machine j has an incentive to move to machine j' iff:

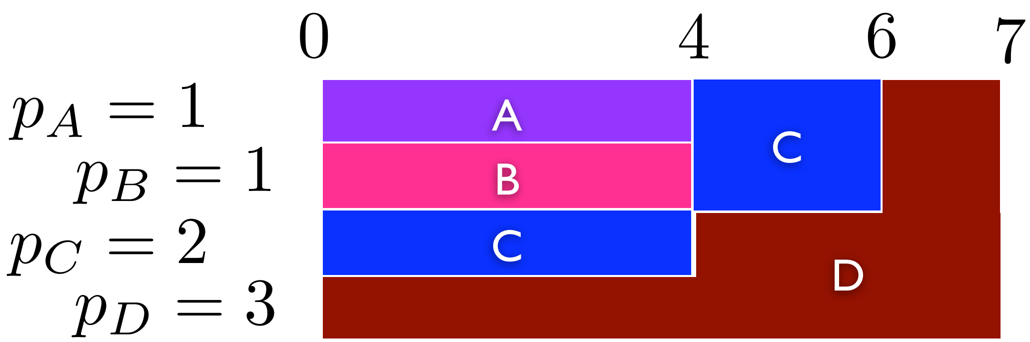
$$p_{ij} + \ell_j > 2p_{ij'} + \ell_{j'}$$

Natural policies

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

Natural policies

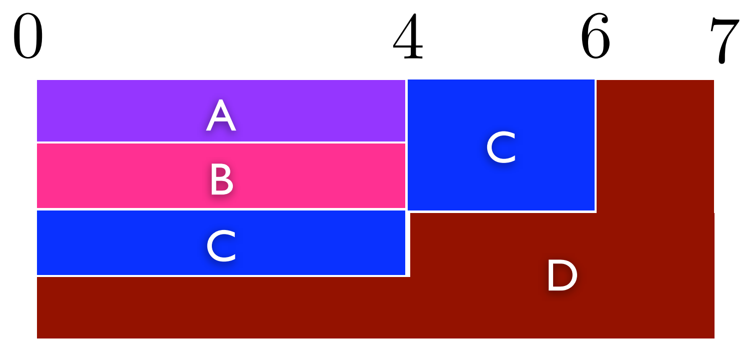
* **EQUI:** schedules jobs in parallel, assigning each job an equal fraction of the processor.



Natural policies

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

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

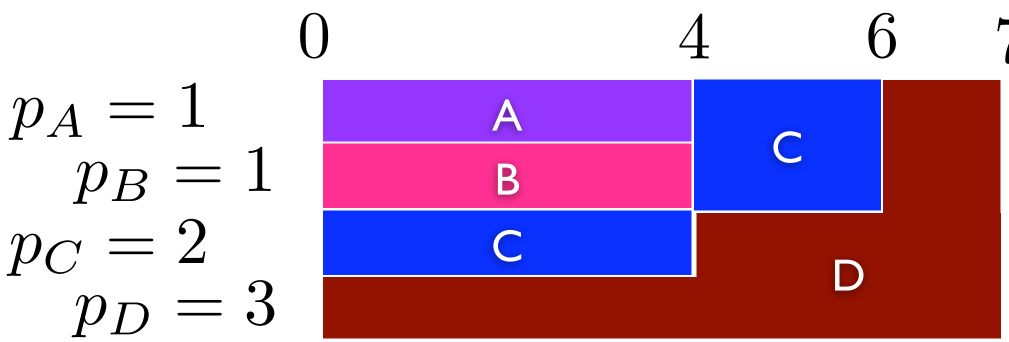


If there are k jobs on machine j s.t: $p_{1j} \leq \ldots \leq p_{kj}$

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

Natural policies

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



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

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

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

sum ⟷ number

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

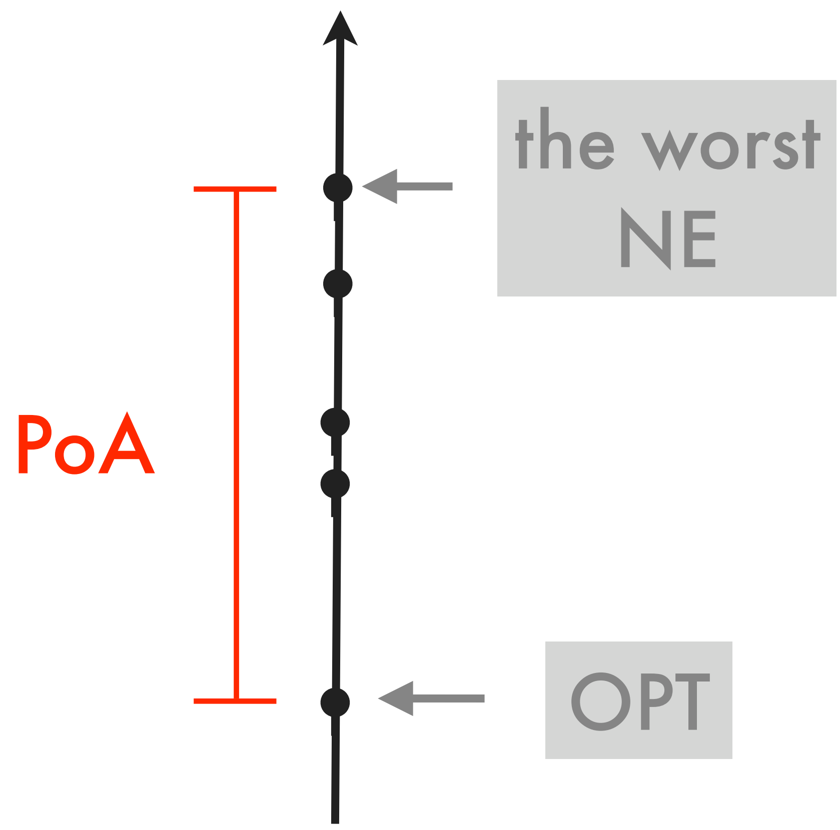
Existence of equilibrium

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

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.



Existence of equilibrium

Standard definitions

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

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

* Def: Best-response dynamic is a process that let an arbitrary unhappy job make a best response.

RANDOM, uniform machines

- Theorem:
  - The game under RANDOM policy is a potential game for uniform machines with balanced jobs (balanced speeds).

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 $s_1 \geq s_2 \geq \ldots \geq s_m$

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

* 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 $s_a > s_b$

RANDOM, uniform machines

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

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

* 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 $s_a > s_b$
* Proof: let i' be a new unhappy job.

RANDOM, uniform machines

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

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

* 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 $s_a > s_b$

* Proof: let i' be a new unhappy job.
  - i' was happy on machine c and now i' has an incentive to move to machine a

RANDOM, uniform machines

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

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

- 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 $s_a > s_b$
   - Proof: let i' be a new unhappy job.
      - i' was happy on machine c and now i' has an incentive to move to machine a
      - i' was happy on machine b and now i' has an incentive to move to machine c

RANDOM, uniform machines

* **Proof:** let i' be a new unhappy job.
  - i' was happy on machine c and now i' has an incentive to move to machine a

$$\ell_a + p_i/s_a > \ell_b + 2p_i/s_b$$

$$\ell_c + p_{i'}/s_c > (\ell_a - p_i/s_a) + 2p_{i'}/s_a$$

$$\ell_c + p_{i'}/s_c \leq \ell_b + 2p_{i'}/s_b$$

RANDOM, uniform machines

- **Proof:** let i' be a new unhappy job.
  - i' was happy on machine c and now i' has an incentive to move to machine a

$$\ell_a + p_i/s_a > \ell_b + 2p_i/s_b$$

$$\ell_c + p_{i'}/s_c > (\ell_a - p_i/s_a) + 2p_{i'}/s_a$$

$$\ell_c + p_{i'}/s_c \leq \ell_b + 2p_{i'}/s_b$$

Hence, $(s_a - s_b)(p_{i'} - p_i) > 0$

RANDOM, uniform machines

* Proof: let i' be a new unhappy job.
  - i' was happy on machine b and now i' has an incentive to move to machine c

$$(p_i - p_{i'})(2s_b - s_c) > 0$$

RANDOM, uniform machines

* Proof: let i' be a new unhappy job.
  - i' was happy on machine b and now i' has an incentive to move to machine c

$$(p_i - p_{i'})(2s_b - s_c) > 0$$

  - The lemma follows.

Potential function

Potential function

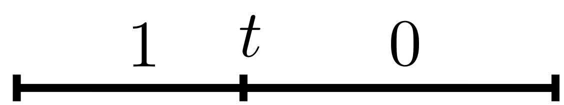
* Dynamic: among all unhappy jobs, let the one with the greatest index make a best move.

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.

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

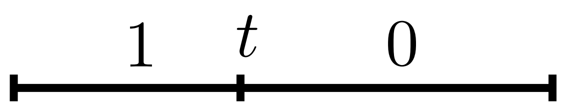


Potential function

* 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.

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



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

Potential function

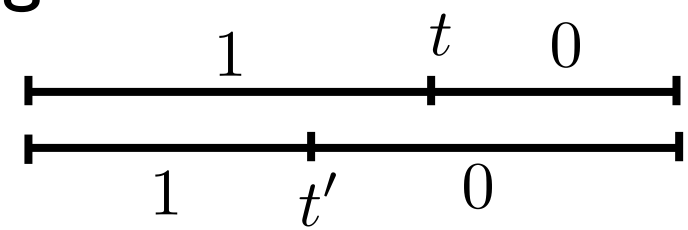
- $\Phi(\sigma) = (f_\sigma(1), s_{\sigma(1)}, \ldots, f_\sigma(n), s_{\sigma(n)})$ lex. decreases

Potential function

* $\Phi(\sigma) = (f_\sigma(1), s_{\sigma(1)}, \ldots, f_\sigma(n), s_{\sigma(n)})$ lex. decreases
* t' is the unhappy player with greatest index in σ'

Potential function

- $\Phi(\sigma) = (f_\sigma(1), s_{\sigma(1)}, \ldots, f_\sigma(n), s_{\sigma(n)})$ lex. decreases
- t' is the unhappy player with greatest index in σ'



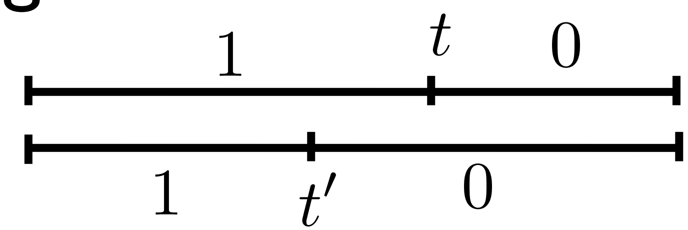
- If $t' < t$

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

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

Potential function

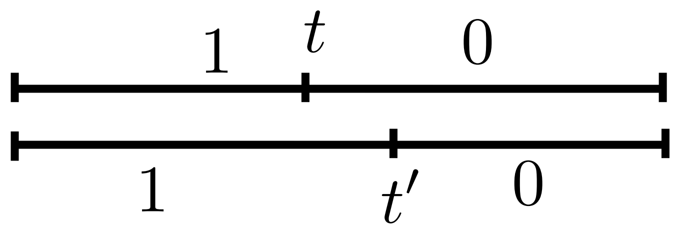
- $\Phi(\sigma) = (f_\sigma(1), s_{\sigma(1)}, \ldots, f_\sigma(n), s_{\sigma(n)})$ lex. decreases
- t' is the unhappy player with greatest index in σ'



- If $t' < t$

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

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



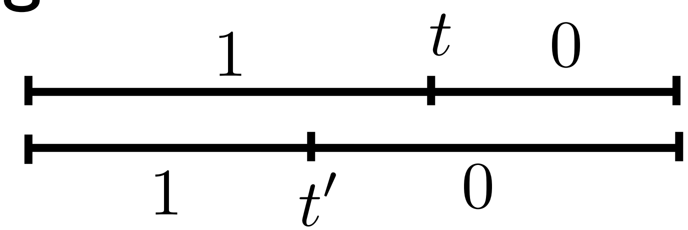
- If $t' > t$

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

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

Potential function

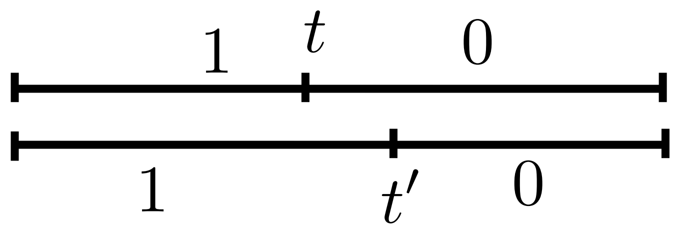
- $\Phi(\sigma) = (f_\sigma(1), s_{\sigma(1)}, \ldots, f_\sigma(n), s_{\sigma(n)})$ lex. decreases
- t' is the unhappy player with greatest index in σ'



- If $t' < t$

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

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



- If $t' > t$

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

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

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

RANDOM, unrel. machines

RANDOM, unrel. machines

- Theorem:
  - The game under RANDOM policy is a potential game for 2 unrelated machines with balanced jobs but it is not for more than 3 machines.

RANDOM, unrel. machines

- Theorem:
  - The game under RANDOM policy is a potential game for 2 unrelated machines with balanced jobs but it is not for more than 3 machines.
- Proof:
  - Let $\sigma : \{1, \ldots, n\} \to \{ 1, 2\}$ be the current profile.
  - The following potential decreases strictly

$$\Phi = |\ell_1 - \ell_2| + 3 \sum_i \max\{p_{i\sigma(i)} - p_{i\overline{\sigma(i)}}, 0\}$$

EQUI

EQUI

* Theorem:
  * The game under EQUI policy is a strong potential game.

EQUI

- Theorem:
  - The game under EQUI policy is a strong potential game.
- Proof:
  - Let σ be the current profile.
  - The following exact potential decreases strictly

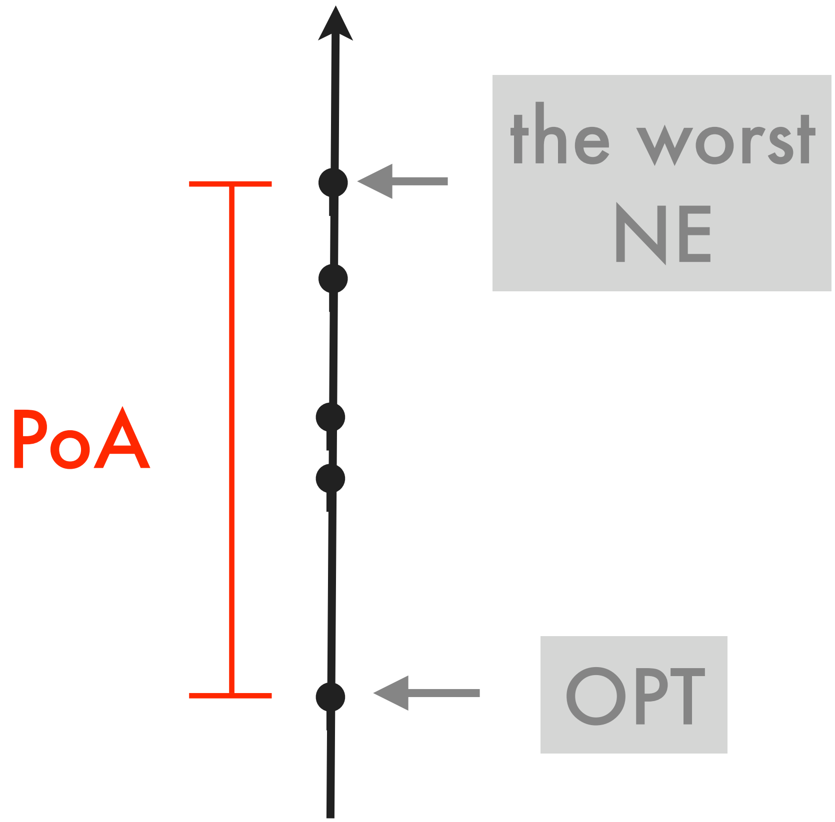
$$\Phi = \frac{1}{2}\sum_i (c_i + p_{i\sigma(i)})$$

Inefficiency of equilibria

Inefficiency

* **Theorem:** For unrelated machines, the PoA of policy EQUI is at most 2m.

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



Proof (sketch)

Proof (sketch)

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

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

Proof (sketch)

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

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

- Proof: $$q_i := \min_j p_{ij}$$ $$\sum_{i=1}^{n} q_i \leq m \cdot OPT$$ $$Q(i) := \arg\min_j p_{ij}$$
  - Rename jobs such that $q_1 \leq q_2 \leq \ldots \leq q_n$

Proof (sketch)

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

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

- **Proof:**

$$q_i := \min_j p_{ij} \qquad \sum_{i=1}^{n} q_i \leq m \cdot OPT$$

$$Q(i) := \arg\min_j p_{ij}$$

  - Rename jobs such that $q_1 \leq q_2 \leq \ldots \leq q_n$
  - **Lemma:** In any NE,

$$c_i \leq 2q_1 + \ldots + 2q_{i-1} + (n-i+1)q_i$$

Proof (sketch)

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

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

* Proof:

$$q_i := \min_j p_{ij} \qquad \sum_{i=1}^{n} q_i \leq m \cdot OPT$$

$$Q(i) := \arg\min_j p_{ij}$$

- Rename jobs such that $q_1 \leq q_2 \leq \ldots \leq q_n$
- Lemma: In any NE,

$$c_i \leq 2q_1 + \ldots + 2q_{i-1} + (n-i+1)q_i$$

- Proof: $c_1 \leq nq_1$ = the worst cost on $Q(1)$

$c_2 \leq q_1 + (n-1)q_2$ = the worst cost on $Q(2)$

Proof (sketch)

- By monotonicity of $(q_i)_{i=1}^n$

$$\text{makespan} = \max_i c_i$$
$$\leq \max_i (2q_1 + \ldots 2q_i + (n-i+1)q_i)$$
$$\leq 2\sum_i q_i$$
$$\leq 2m \cdot OPT$$

$$PoA \leq 2m$$

Lower bound

- Theorem: The strong PoA of EQUI is at least (m+1)/4.

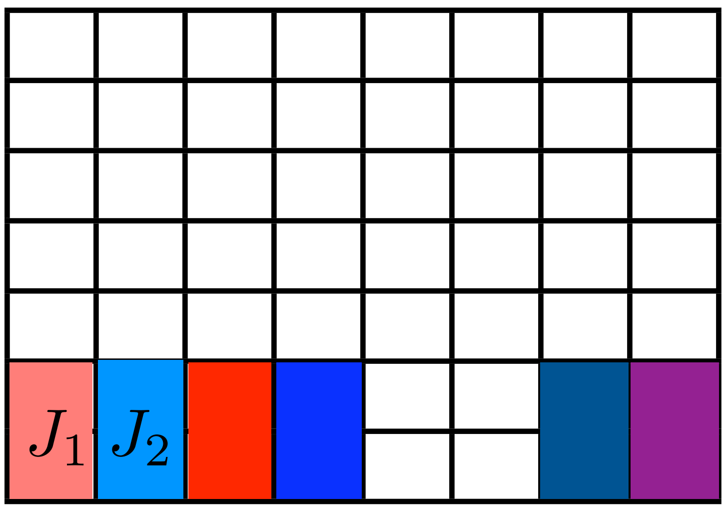
Lower bound

* Theorem: The strong PoA of EQUI is at least (m+1)/4.
* Proof (idea):
  - m groups of jobs $J_1, \ldots, J_m$
  - Jobs in J_i can be scheduled only on machines $i, i+1$

Lower bound

- **Theorem:** The strong PoA of EQUI is at least (m+1)/4.
- **Proof (idea):**
  - m groups of jobs $J_1, \ldots, J_m$
  - Jobs in J_i can be scheduled only on machines $i, i+1$

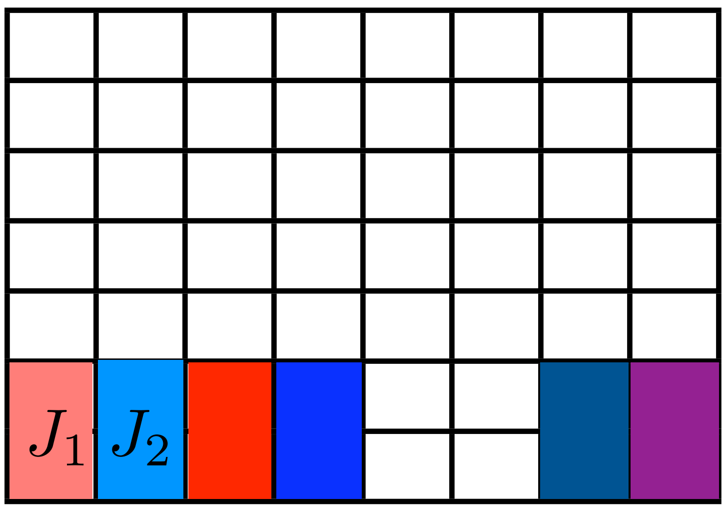
OPT



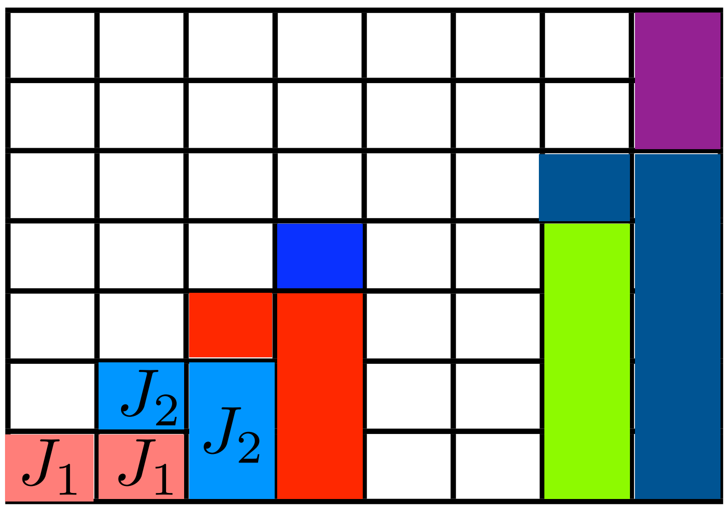
Lower bound

- **Theorem:** The strong PoA of EQUI is at least (m+1)/4.
- **Proof (idea):**
  - m groups of jobs $J_1, \ldots, J_m$
  - Jobs in J_i can be scheduled only on machines $i, i+1$

OPT



NE



Conclusion

☑ Knowledge of jobs' characteristics is not necessarily needed while restricting to strongly local policies.

Conclusion

☑ Knowledge of jobs' characteristics is not necessarily needed while restricting to strongly local policies.

☐ Study the existence of equilibrium for RANDOM in two unrelated machines and in uniform machines.

Conclusion

☑ Knowledge of jobs' characteristics is not necessarily needed while restricting to strongly local policies.

☐ Study the existence of equilibrium for RANDOM in two unrelated machines and in uniform machines.

☐ Designing local policy with PoA = $o(\log m)$