Coordination Mechanisms for Weighted Sum of Completion Times in Machine Scheduling

Vasilis Gkatzelis

Courant Institute, New York University

ACAC 2010

Joint work with:

Richard Cole — Courant Institute, New York University
Vahab Mirrokni — Google Research, New York

Outline

1. Machine Scheduling
   - Model
   - Previous Results
2. Selfish Machine Scheduling
   - Model
   - Relevant Results
3. SmithRule
   - Robust PoA Bound
4. ProportionalSharing
   - Sum of Completion Times
   - Exact Potential Games
   - Robust PoA Bound
5. Approximation Algorithm

Machine Scheduling

- We have a set N of n jobs and a set M of m machines
- Each job needs to be assigned to exactly one machine
- Each machine can process only one job at any time
- A schedule defines which job will be processed by each machine at any point
- For each job i, we use the following notation:
  - It's processing time on machine j is denoted by p_{ij}
  - It's weight is denoted by w_i
  - It's completion time under a specific schedule is denoted by c_i

Machine Scheduling

- We have a set N of n jobs and a set M of m machines
- Each job needs to be assigned to exactly one machine
- Each machine can process only one job at any time
- A schedule defines which job will be processed by each machine at any point
- For each job i, we use the following notation:
  - It's processing time on machine j is denoted by p_{ij}
  - It's weight is denoted by w_i
  - It's completion time under a specific schedule is denoted by c_i

Machine Scheduling

- We have a set N of n jobs and a set M of m machines
- Each job needs to be assigned to exactly one machine
- Each machine can process only one job at any time
- A schedule defines which job will be processed by each machine at any point
- For each job i, we use the following notation:
  - It's processing time on machine j is denoted by p_{ij}
  - It's weight is denoted by w_i
  - It's completion time under a specific schedule is denoted by c_i

Machine Scheduling

- We have a set N of n jobs and a set M of m machines
- Each job needs to be assigned to exactly one machine
- Each machine can process only one job at any time
- A schedule defines which job will be processed by each machine at any point
- For each job i, we use the following notation:
  - It's processing time on machine j is denoted by p_{ij}
  - It's weight is denoted by w_i
  - It's completion time under a specific schedule is denoted by c_i

Machine Scheduling

- We have a set N of n jobs and a set M of m machines
- Each job needs to be assigned to exactly one machine
- Each machine can process only one job at any time
- A schedule defines which job will be processed by each machine at any point
- For each job i, we use the following notation:
  - It's processing time on machine j is denoted by p_{ij}
  - It's weight is denoted by w_i
  - It's completion time under a specific schedule is denoted by c_i

Machine Scheduling

- We have a set N of n jobs and a set M of m machines
- Each job needs to be assigned to exactly one machine
- Each machine can process only one job at any time
- A schedule defines which job will be processed by each machine at any point
- For each job i, we use the following notation:
  - It's processing time on machine j is denoted by p_{ij}
  - It's weight is denoted by w_i
  - It's completion time under a specific schedule is denoted by c_i

Machine Scheduling

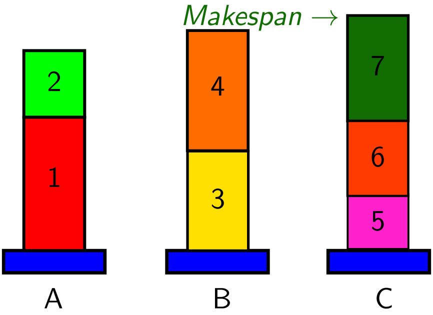
- We have a set N of n jobs and a set M of m machines
- Each job needs to be assigned to exactly one machine
- Each machine can process only one job at any time
- A schedule defines which job will be processed by each machine at any point
- For each job i, we use the following notation:
  - It's processing time on machine j is denoted by p_{ij}
  - It's weight is denoted by w_i
  - It's completion time under a specific schedule is denoted by c_i

Machine Scheduling

- We have a set N of n jobs and a set M of m machines
- Each job needs to be assigned to exactly one machine
- Each machine can process only one job at any time
- A schedule defines which job will be processed by each machine at any point
- For each job i, we use the following notation:
  - It's processing time on machine j is denoted by p_{ij}
  - It's weight is denoted by w_i
  - It's completion time under a specific schedule is denoted by c_i

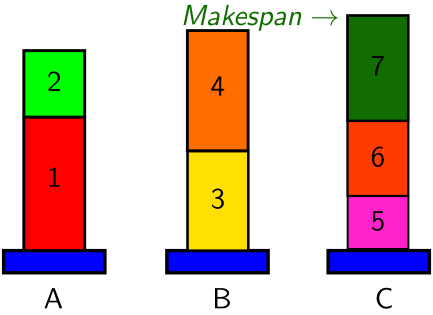
Objective Functions

- Makespan ($\max_i c_i$)
- Sum of completion times ($\sum_i c_i$)
- Weighted sum of completion times ($\sum_i w_i c_i$)



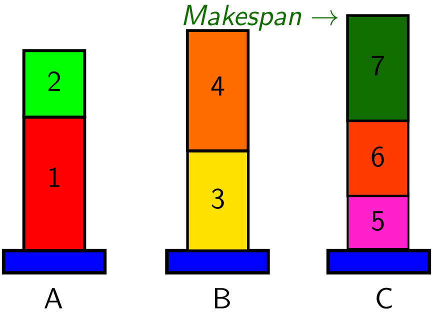
Objective Functions

- Makespan ($\max_i c_i$)
- Sum of completion times ($\sum_i c_i$)
- Weighted sum of completion times ($\sum_i w_i c_i$)



Objective Functions

- Makespan ($\max_i c_i$)
- Sum of completion times ($\sum_i c_i$)
- Weighted sum of completion times ($\sum_i w_i c_i$)



Machine Models

- Identical machines
  - Each job i has a processing requirement p_i
  - The processing time of job i on any machine j will be $p_{ij} = p_i$
- Related machines
  - Each job i has a processing requirement p_i
  - Each machine j has a speed q_j
  - The processing time of job i on machine j will then be $p_{ij} = \frac{p_i}{q_j}$
- Restricted machines
  - Each job i has a processing requirement p_i
  - The processing time of job i on machine j will either be $p_{ij} = p_i$ or $p_{ij} = \infty$
- Unrelated machines
  - The processing time of job i on machine j can be arbitrary

Machine Models

- Identical machines
  - Each job i has a processing requirement p_i
  - The processing time of job i on any machine j will be $p_{ij} = p_i$
- Related machines
  - Each job i has a processing requirement p_i
  - Each machine j has a speed q_j
  - The processing time of job i on machine j will then be $p_{ij} = \frac{p_i}{q_j}$
- Restricted machines
  - Each job i has a processing requirement p_i
  - The processing time of job i on machine j will either be $p_{ij} = p_i$ or $p_{ij} = \infty$
- Unrelated machines
  - The processing time of job i on machine j can be arbitrary

Machine Models

- Identical machines
  - Each job i has a processing requirement p_i
  - The processing time of job i on any machine j will be $p_{ij} = p_i$
- Related machines
  - Each job i has a processing requirement p_i
  - Each machine j has a speed q_j
  - The processing time of job i on machine j will then be $p_{ij} = \frac{p_i}{q_j}$
- Restricted machines
  - Each job i has a processing requirement p_i
  - The processing time of job i on machine j will either be $p_{ij} = p_i$ or $p_{ij} = \infty$
- Unrelated machines
  - The processing time of job i on machine j can be arbitrary

Machine Models

- Identical machines
  - Each job i has a processing requirement p_i
  - The processing time of job i on any machine j will be $p_{ij} = p_i$
- Related machines
  - Each job i has a processing requirement p_i
  - Each machine j has a speed q_j
  - The processing time of job i on machine j will then be $p_{ij} = \frac{p_i}{q_j}$
- Restricted machines
  - Each job i has a processing requirement p_i
  - The processing time of job i on machine j will either be $p_{ij} = p_i$ or $p_{ij} = \infty$
- Unrelated machines
  - The processing time of job i on machine j can be arbitrary

Machine Scheduling

- Objective functions:
  - Makespan ($\max_i c_i$)
  - Sum of completion times ($\sum_i c_i$)
  - Weighted sum of completion times ($\sum_i w_i c_i$)
- Machine models:
  - identical machines
  - related machines
  - restricted machines
  - unrelated machines

Machine Scheduling

- Objective functions:
  - Makespan ($\max_i c_i$)
  - Sum of completion times ($\sum_i c_i$)
  - Weighted sum of completion times ($\sum_i w_i c_i$)
- Machine models:
  - identical machines
  - related machines
  - restricted machines
  - unrelated machines

Previous Results

- Minimizing $\sum_i c_i$ is in P even for unrelated machines [H 73, BCS 74]
- Minimizing $\sum_i w_i c_i$ is NP-hard even for identical machines [LKB 77]
  - For identical machines there exists a PTAS [SW 00]
  - For unrelated machines the problem is APX-hard [HSW 98]
  - Constant factor approximation algorithms...
- New: *Combinatorial* constant factor approximation algorithm

Previous Results

- Minimizing $\sum_i c_i$ is in P even for unrelated machines [H 73, BCS 74]
- Minimizing $\sum_i w_i c_i$ is NP-hard even for identical machines [LKB 77]
  - For identical machines there exists a PTAS [SW 00]
  - For unrelated machines the problem is APX-hard [HSW 98]
  - Constant factor approximation algorithms...
- New: *Combinatorial* constant factor approximation algorithm

Previous Results

- Minimizing $\sum_i c_i$ is in P even for unrelated machines [H 73, BCS 74]
- Minimizing $\sum_i w_i c_i$ is NP-hard even for identical machines [LKB 77]
  - For identical machines there exists a PTAS [SW 00]
  - For unrelated machines the problem is APX-hard [HSW 98]
  - Constant factor approximation algorithms...
- New: *Combinatorial* constant factor approximation algorithm

Previous Results

- Minimizing $\sum_i c_i$ is in P even for unrelated machines [H 73, BCS 74]
- Minimizing $\sum_i w_i c_i$ is NP-hard even for identical machines [LKB 77]
  - For identical machines there exists a PTAS [SW 00]
  - For unrelated machines the problem is APX-hard [HSW 98]
  - Constant factor approximation algorithms...
- New: *Combinatorial* constant factor approximation algorithm

Previous Results

- Minimizing $\sum_i c_i$ is in P even for unrelated machines [H 73, BCS 74]
- Minimizing $\sum_i w_i c_i$ is NP-hard even for identical machines [LKB 77]
  - For identical machines there exists a PTAS [SW 00]
  - For unrelated machines the problem is APX-hard [HSW 98]
  - Constant factor approximation algorithms...
- New: *Combinatorial* constant factor approximation algorithm

Previous Results

- Minimizing $\sum_i c_i$ is in P even for unrelated machines [H 73, BCS 74]
- Minimizing $\sum_i w_i c_i$ is NP-hard even for identical machines [LKB 77]
  - For identical machines there exists a PTAS [SW 00]
  - For unrelated machines the problem is APX-hard [HSW 98]
  - Constant factor approximation algorithms...
- New: *Combinatorial* constant factor approximation algorithm

Selfish Machine Scheduling

- Let each job be controlled by a selfish agent
- Each agent's strategy set is the set of machines
- Given a strategy choice s_i for each player i, we get an assignment s of jobs to machines
- The cost that each player will incur (it's completion time), given s, depends on the machines' policies

Selfish Machine Scheduling

- Let each job be controlled by a selfish agent
- Each agent's strategy set is the set of machines
- Given a strategy choice s_i for each player i, we get an assignment s of jobs to machines
- The cost that each player will incur (it's completion time), given s, depends on the machines' policies

Selfish Machine Scheduling

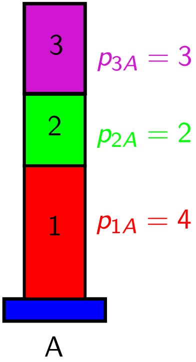
- Let each job be controlled by a selfish agent
- Each agent's strategy set is the set of machines
- Given a strategy choice s_i for each player i, we get an assignment s of jobs to machines
- The cost that each player will incur (it's completion time), given s, depends on the machines' policies

Selfish Machine Scheduling

- Let each job be controlled by a selfish agent
- Each agent's strategy set is the set of machines
- Given a strategy choice s_i for each player i, we get an assignment s of jobs to machines
- The cost that each player will incur (it's completion time), given s, depends on the machines' policies

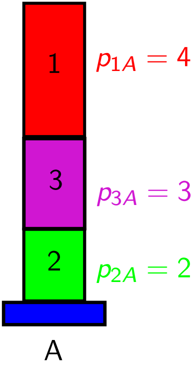
Strongly Local Policies

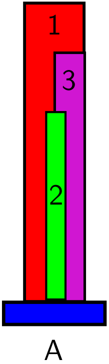
For example ShortestFirst and EqualSharing:



Strongly Local Policies

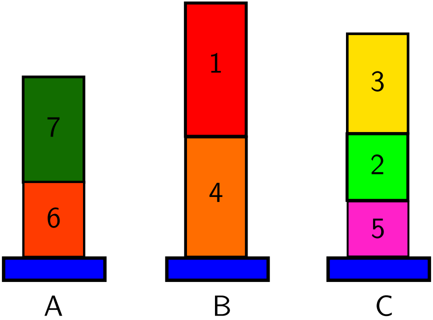
For example ShortestFirst and EqualSharing:





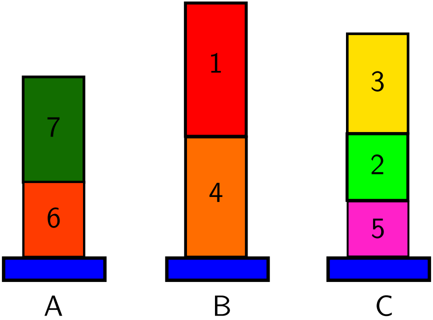
Coordination Mechanism

- By Christodoulou, Koutsoupias and Nanavati [ICALP 04]
- A set of local policies, one for each machine



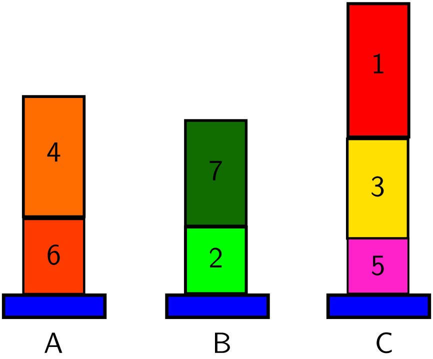
Coordination Mechanism

- By Christodoulou, Koutsoupias and Nanavati [ICALP 04]
- A set of local policies, one for each machine



Coordination Mechanism

- By Christodoulou, Koutsoupias and Nanavati [ICALP 04]
- A set of local policies, one for each machine



Normal Form Game

- Assume that all the job weights are equal to 1
- Given a coordination mechanism α we have defined a game
- Each assignment (strategy profile) s implies a completion time or cost denoted by $c_i^\alpha(s)$ for each player i
- An assignment s is a Pure Nash Equilibrium (PNE) if:

$$\forall i \in N, \forall s_i' \in M, \quad c_i^\alpha(s_{-i}, s_i') \geq c_i^\alpha(s)$$

Normal Form Game

- Assume that all the job weights are equal to 1
- Given a coordination mechanism α we have defined a game
- Each assignment (strategy profile) s implies a completion time or cost denoted by $c_i^\alpha(s)$ for each player i
- An assignment s is a Pure Nash Equilibrium (PNE) if:

$$\forall i \in N, \forall s_i' \in M, \quad c_i^\alpha(s_{-i}, s_i') \geq c_i^\alpha(s)$$

Normal Form Game

- Assume that all the job weights are equal to 1
- Given a coordination mechanism α we have defined a game
- Each assignment (strategy profile) s implies a completion time or cost denoted by $c_i^\alpha(s)$ for each player i
- An assignment s is a Pure Nash Equilibrium (PNE) if:

$$\forall i \in N, \forall s_i' \in M, \quad c_i^\alpha(s_{-i}, s_i') \geq c_i^\alpha(s)$$

Normal Form Game

- Assume that all the job weights are equal to 1
- Given a coordination mechanism α we have defined a game
- Each assignment (strategy profile) s implies a completion time or cost denoted by $c_i^\alpha(s)$ for each player i
- An assignment s is a Pure Nash Equilibrium (PNE) if:

$$\forall i \in N, \forall s_i' \in M, \;\; c_i^\alpha(s_{-i}, s_i') \geq c_i^\alpha(s)$$

Price of Anarchy

- Defined by Koutsoupias and Papadimitriou [STACS 99]
- The PoA of the induced game w.r.t. the sum of completion times is:

$$\max_{s \in \text{PNE}} \frac{\sum_{i \in N} c_i^{\alpha}(s)}{\sum_{i \in N} c_i^{\alpha}(s^{\alpha})}$$

- The PoA of the **coordination mechanism** w.r.t. the sum of completion times is:

$$\max_{s \in \text{PNE}} \frac{\sum_{i \in N} c_i^{\alpha}(s)}{\sum_{i \in N} c_i^{\text{SF}}(s^{*})}$$

Price of Anarchy

- Defined by Koutsoupias and Papadimitriou [STACS 99]
- The PoA of the induced game w.r.t. the sum of completion times is:

$$\max_{s \in \text{PNE}} \frac{\sum_{i \in N} c_i^{\alpha}(s)}{\sum_{i \in N} c_i^{\alpha}(s^{\alpha})}$$

- The PoA of the **coordination mechanism** w.r.t. the sum of completion times is:

$$\max_{s \in \text{PNE}} \frac{\sum_{i \in N} c_i^{\alpha}(s)}{\sum_{i \in N} c_i^{\mathbf{SF}}(s^*)}$$

Price of Anarchy

- Defined by Koutsoupias and Papadimitriou [STACS 99]
- The PoA of the induced game w.r.t. the sum of completion times is:

$$\max_{s \in \text{PNE}} \frac{\sum_{i \in N} c_i^{\alpha}(s)}{\sum_{i \in N} c_i^{\alpha}(s^{\alpha})}$$

- The PoA of the **coordination mechanism** w.r.t. the sum of completion times is:

$$\max_{s \in \text{PNE}} \frac{\sum_{i \in N} c_i^{\alpha}(s)}{\sum_{i \in N} c_i^{\mathbf{SF}}(s^{*})}$$

- LongestFirst policy for identical machines [CKN 04]
- Then [ILMS 05]
  - Study and survey results for four coordination mechanisms and four machine models
  - Study convergence time and existence of PNE
- Next [AJM 08]:
  - Prove that strongly local ordering policies are $\Omega(m)$
  - Present a local policy that achieves $O(\log m)$ but doesn't induce potential games
  - Present a local policy that achieves $O(\log^2 m)$ and induces potential games

- LongestFirst policy for identical machines [CKN 04]
- Then [ILMS 05]
  - Study and survey results for four coordination mechanisms and four machine models
  - Study convergence time and existence of PNE
- Next [AJM 08]:
  - Prove that strongly local ordering policies are $\Omega(m)$
  - Present a local policy that achieves $O(\log m)$ but doesn't induce potential games
  - Present a local policy that achieves $O(\log^2 m)$ and induces potential games

- LongestFirst policy for identical machines [CKN 04]
- Then [ILMS 05]
  - Study and survey results for four coordination mechanisms and four machine models
  - Study convergence time and existence of PNE
- Next [AJM 08]:
  - Prove that strongly local ordering policies are $\Omega(m)$
  - Present a local policy that achieves $O(\log m)$ but doesn't induce potential games
  - Present a local policy that achieves $O(\log^2 m)$ and induces potential games

- Eventually [C 09]:
  - Presents three new coordination mechanisms for unrelated machines
  - One of those mechanisms achieves $O(\log m)$ and induces potential games
  - Another one of those mechanisms is preemptive and achieves $O(\log m/\log\log m)$
- A lower bound of $\Omega(\log m)$ for all local ordering policies was presented by [FS 10]
- EqualSharing induces potential games and has PoA $\Theta(m)$ [DT 09]
- SmithRule for related restricted machines and PNE [CQ 10]

- Eventually [C 09]:
  - Presents three new coordination mechanisms for unrelated machines
  - One of those mechanisms achieves $O(\log m)$ and induces potential games
  - Another one of those mechanisms is preemptive and achieves $O(\log m / \log \log m)$
- A lower bound of $\Omega(\log m)$ for all local ordering policies was presented by [FS 10]
- EqualSharing induces potential games and has PoA $\Theta(m)$ [DT 09]
- SmithRule for related restricted machines and PNE [CQ 10]

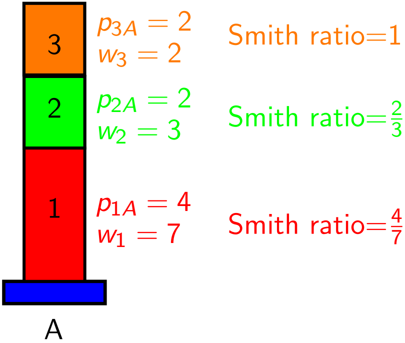
- Eventually [C 09]:
  - Presents three new coordination mechanisms for unrelated machines
  - One of those mechanisms achieves $O(\log m)$ and induces potential games
  - Another one of those mechanisms is preemptive and achieves $O(\log m / \log\log m)$
- A lower bound of $\Omega(\log m)$ for all local ordering policies was presented by [FS 10]
- EqualSharing induces potential games and has PoA $\Theta(m)$ [DT 09]
- SmithRule for related restricted machines and PNE [CQ 10]

- Eventually [C 09]:
  - Presents three new coordination mechanisms for unrelated machines
  - One of those mechanisms achieves $O(\log m)$ and induces potential games
  - Another one of those mechanisms is preemptive and achieves $O(\log m / \log \log m)$
- A lower bound of $\Omega(\log m)$ for all local ordering policies was presented by [FS 10]
- EqualSharing induces potential games and has PoA $\Theta(m)$ [DT 09]
- SmithRule for related restricted machines and PNE [CQ 10]

Outline

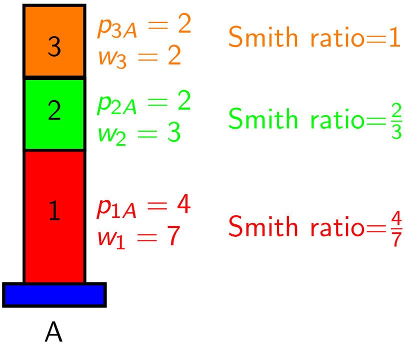
SmithRule policy

- Non-preemptive policy
- Each machine j gives higher priority to jobs with smaller $\frac{p_{ij}}{w_i}$
- In the optimal solution, every machine must follow this policy



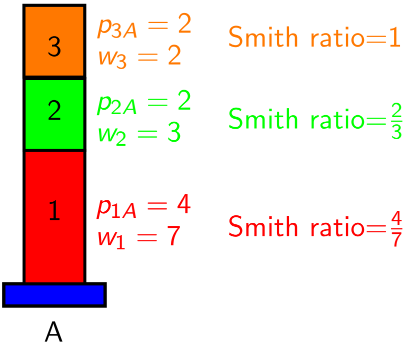
SmithRule policy

- Non-preemptive policy
- Each machine j gives higher priority to jobs with smaller $\frac{p_{ij}}{w_i}$
- In the optimal solution, every machine must follow this policy



SmithRule policy

- Non-preemptive policy
- Each machine j gives higher priority to jobs with smaller $\frac{p_{ij}}{w_i}$
- In the optimal solution, every machine must follow this policy



PoA for weighted sum of completion times

- An assignment s is a Pure Nash Equilibrium if:

$$\forall i \in N, \forall s_i' \in M, \quad w_i c_i^{\alpha}(s_{-i}, s_i') \geq w_i c_i^{\alpha}(s)$$

- The PoA of the induced game w.r.t. the weighted sum of completion times is: $\max_{s \in \text{PNE}} \frac{\sum_{i \in N} w_i c_i^{\alpha}(s)}{\sum_{i \in N} w_i c_i^{\alpha}(s^{\alpha})}$
- The PoA of the coordination mechanism w.r.t. the weighted sum of completion times is:

$$\max_{s \in \text{PNE}} \frac{\sum_{i \in N} w_i c_i^{\alpha}(s)}{\sum_{i \in N} w_i c_i^{\text{SR}}(s^*)} = \max_{s \in \text{PNE}} \frac{\sum_{i \in N} w_i c_i^{\alpha}(s)}{OPT}$$

PoA for weighted sum of completion times

- An assignment s is a Pure Nash Equilibrium if:

$$\forall i \in N, \forall s_i' \in M, \;\; w_i c_i^{\alpha}(s_{-i}, s_i') \geq w_i c_i^{\alpha}(s)$$

- The PoA of the induced game w.r.t. the weighted sum of completion times is: $\max_{s \in \text{PNE}} \frac{\sum_{i \in N} w_i c_i^{\alpha}(s)}{\sum_{i \in N} w_i c_i^{\alpha}(s^{\alpha})}$
- The PoA of the coordination mechanism w.r.t. the weighted sum of completion times is:

$$\max_{s \in \text{PNE}} \frac{\sum_{i \in N} w_i c_i^{\alpha}(s)}{\sum_{i \in N} w_i c_i^{\mathbf{SR}}(s^*)} = \max_{s \in \text{PNE}} \frac{\sum_{i \in N} w_i c_i^{\alpha}(s)}{OPT}$$

PoA for weighted sum of completion times

- An assignment s is a Pure Nash Equilibrium if:

$$\forall i \in N, \forall s_i' \in M, \quad w_i c_i^{\alpha}(s_{-i}, s_i') \geq w_i c_i^{\alpha}(s)$$

- The PoA of the induced game w.r.t. the weighted sum of completion times is: $\max_{s \in \text{PNE}} \frac{\sum_{i \in N} w_i c_i^{\alpha}(s)}{\sum_{i \in N} w_i c_i^{\alpha}(s^{\alpha})}$
- The PoA of the coordination mechanism w.r.t. the weighted sum of completion times is:

$$\max_{s \in \text{PNE}} \frac{\sum_{i \in N} w_i c_i^{\alpha}(s)}{\sum_{i \in N} w_i c_i^{\mathbf{SR}}(s^*)} = \max_{s \in \text{PNE}} \frac{\sum_{i \in N} w_i c_i^{\alpha}(s)}{OPT}$$

Robust PoA

- Defined by Roughgarden [STOC 09]

A coordination mechanism α is defined to be (λ, μ)-smooth if for every two assignments s and s^* of any game that it may induce

$$\sum_{i \in N} w_i c_i^{\alpha}(s_{-i}, s_i^*) \leq \lambda \sum_{i \in N} w_i c_i^{\mathsf{SR}}(s^*) + \mu \sum_{i \in N} w_i c_i^{\alpha}(s).$$

Definition

The *Robust PoA* of a coordination mechanism is equal to $\inf\left\{\frac{\lambda}{1-\mu} : (\lambda, \mu) \text{ s.t. the mechanism is } (\lambda, \mu)\text{-smooth}\right\}$.

Robust PoA

- Defined by Roughgarden [STOC 09]

A coordination mechanism α is defined to be (λ, μ)-smooth if for every two assignments s and s^* of any game that it may induce

$$\sum_{i \in N} w_i c_i^{\alpha}(s_{-i}, s_i^*) \leq \lambda \sum_{i \in N} w_i c_i^{\mathbf{SR}}(s^*) + \mu \sum_{i \in N} w_i c_i^{\alpha}(s).$$

Definition

The *Robust PoA* of a coordination mechanism is equal to $\inf\left\{\frac{\lambda}{1-\mu} : (\lambda, \mu) \text{ s.t. the mechanism is } (\lambda, \mu)\text{-smooth}\right\}$.

Robust PoA

- Defined by Roughgarden [STOC 09]

A coordination mechanism α is defined to be (λ, μ)-smooth if for every two assignments s and s^* of any game that it may induce

$$\sum_{i \in N} w_i c_i^{\alpha}(s_{-i}, s_i^*) \leq \lambda \sum_{i \in N} w_i c_i^{\mathbf{SR}}(s^*) + \mu \sum_{i \in N} w_i c_i^{\alpha}(s).$$

Definition

The *Robust PoA* of a coordination mechanism is equal to $\inf \left\{ \frac{\lambda}{1-\mu} : (\lambda, \mu) \text{ s.t. the mechanism is } (\lambda, \mu)\text{-smooth} \right\}$.

Theorem

The Robust PoA of SmithRule for unrelated machines is at most 4.

We show that this coordination mechanism is $(2, \frac{1}{2})$-smooth by showing that for any two assignments s and s^*:

$$\sum_{i \in N} w_i c_i^{SR}(s_{-i}, s_i^*) \leq 2C^{SR}(s^*) + \frac{1}{2}C^{SR}(s).$$

Theorem

The Robust PoA of SmithRule for unrelated machines is at most 4.

We show that this coordination mechanism is $(2, \frac{1}{2})$-smooth by showing that for any two assignments s and s^*:

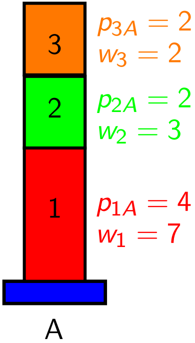
$$\sum_{i \in N} w_i c_i^{SR}(s_{-i}, s_i^*) \leq 2C^{SR}(s^*) + \frac{1}{2}C^{SR}(s).$$

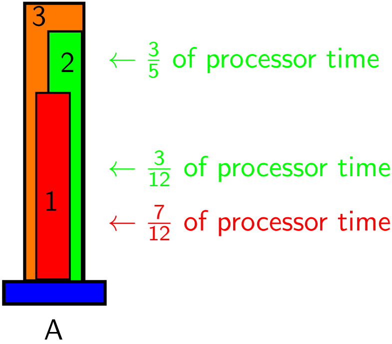
Theorem

The pure PoA of *any set of strongly local ordering policies* for restriced identical machines is at least 4. This is true even for the unweighted case. (Generalizing [CFKKM 06] and [CQ 10])

Proportional Sharing

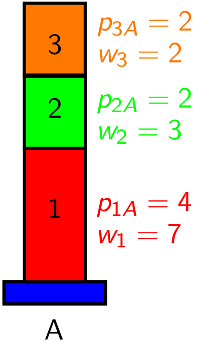
- A generalization of the EqualSharing policy for weighted jobs
- Each job gets a share of the processing time equal to the ratio of its weight over the sum of the weights of all jobs being processed on the same machine at that time

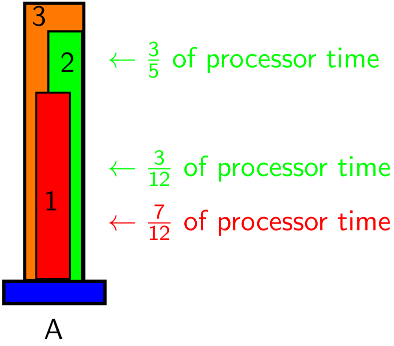




Proportional Sharing

- A generalization of the EqualSharing policy for weighted jobs
- Each job gets a share of the processing time equal to the ratio of its weight over the sum of the weights of all jobs being processed on the same machine at that time





EqualSharing

Theorem

The robust PoA of EqualSharing for unrelated machines is at most 2.5.
This bound is tight even for the restricted related machines model [CFKKM 06].

We show that this coordination mechanism is $(5/3, 1/3)$-smooth by showing that for any two assignments s and s^*:

$$\sum_{i \in N} c_i^{ES}(s_{-i}, s_i^*) \leq \frac{5}{3} \sum_{i \in N} c_i^{SF}(s^*) + \frac{1}{3} \sum_{i \in N} c_i^{ES}(s).$$

EqualSharing

Theorem

The robust PoA of EqualSharing for unrelated machines is at most 2.5.
This bound is tight even for the restricted related machines model [CFKKM 06].

We show that this coordination mechanism is $(5/3, 1/3)$-smooth by showing that for any two assignments s and s^*:

$$\sum_{i \in N} c_i^{ES}(s_{-i}, s_i^*) \leq \frac{5}{3} \sum_{i \in N} c_i^{SF}(s^*) + \frac{1}{3} \sum_{i \in N} c_i^{ES}(s).$$

Theorem

The ProportionalSharing coordination mechanism induces exact potential games.

We show that $\Phi(s) = \frac{1}{2}\sum_{i' \in N} w_{i'} \left(c_{i'}(s) + p_{i' s_{i'}} \right)$ serves as an exact potential function for these games.

Theorem

The ProportionalSharing coordination mechanism induces exact potential games.

We show that $\Phi(s) = \frac{1}{2}\sum_{i' \in N} w_{i'} \left(c_{i'}(s) + p_{i's_{i'}}\right)$ serves as an exact potential function for these games.

Theorem

The robust PoA of ProportionalSharing for unrelated machines is at most $\phi + 1 = \frac{3+\sqrt{5}}{2} \approx 2.618$.
This bound is tight even for the restricted related machines model [CFKKM 06].

We show that this coordination mechanism is $\left(\frac{\phi+2}{2}, \frac{1}{2\phi}\right)$-smooth by showing that for any two assignments s and s^*:

$$\sum_{i \in N} w_i c_i^{PS}(s_{-i}, s_i^*) \leq \frac{\phi + 2}{2} C^{SR}(s^*) + \frac{1}{2\phi} C^{PS}(s).$$

Theorem

The robust PoA of ProportionalSharing for unrelated machines is at most $\phi+1=\frac{3+\sqrt{5}}{2}\approx 2.618$.
This bound is tight even for the restricted related machines model [CFKKM 06].

We show that this coordination mechanism is $\left(\frac{\phi+2}{2},\frac{1}{2\phi}\right)$-smooth by showing that for any two assignments s and s^*:

$$\sum_{i\in N} w_i c_i^{PS}(s_{-i},s_i^*) \leq \frac{\phi+2}{2}C^{SR}(s^*)+\frac{1}{2\phi}C^{PS}(s).$$

Approximation Algorithm

- For unrelated machines and weighted sum of completion times, the minimization problem is NP-hard [LKB 77]
- First constant factor ($\frac{16}{3}$) approximation algorithm [HSSW 97]
- Later improved to $\frac{3}{2} + \epsilon$ [SS 02]
- Independently further improved to $\frac{3}{2}$ by [SS 99, S 01]

The only known constant factor approximation algorithms are based on LP or convex quadratic program relaxations!

Approximation Algorithm

- For unrelated machines and weighted sum of completion times, the minimization problem is NP-hard [LKB 77]
- First constant factor ($\frac{16}{3}$) approximation algorithm [HSSW 97]
- Later improved to $\frac{3}{2} + \epsilon$ [SS 02]
- Independently further improved to $\frac{3}{2}$ by [SS 99, S 01]

The only known constant factor approximation algorithms are based on LP or convex quadratic program relaxations!

Approximation Algorithm

- For unrelated machines and weighted sum of completion times, the minimization problem is NP-hard [LKB 77]
- First constant factor ($\frac{16}{3}$) approximation algorithm [HSSW 97]
- Later improved to $\frac{3}{2} + \epsilon$ [SS 02]
- Independently further improved to $\frac{3}{2}$ by [SS 99, S 01]

The only known constant factor approximation algorithms are based on LP or convex quadratic program relaxations!

Approximation Algorithm

- For unrelated machines and weighted sum of completion times, the minimization problem is NP-hard [LKB 77]
- First constant factor ($\frac{16}{3}$) approximation algorithm [HSSW 97]
- Later improved to $\frac{3}{2} + \epsilon$ [SS 02]
- Independently further improved to $\frac{3}{2}$ by [SS 99, S 01]

The only known constant factor approximation algorithms are based on LP or convex quadratic program relaxations!

Approximation Algorithm

- For unrelated machines and weighted sum of completion times, the minimization problem is NP-hard [LKB 77]
- First constant factor ($\frac{16}{3}$) approximation algorithm [HSSW 97]
- Later improved to $\frac{3}{2} + \epsilon$ [SS 02]
- Independently further improved to $\frac{3}{2}$ by [SS 99, S 01]

The only known constant factor approximation algorithms are based on LP or convex quadratic program relaxations!

Approximation Algorithm

- We know that a PNE always exists for ProportionalSharing
- For any such PNE s we know that $\sum_i w_i c_i^{PS}(s) \leq 2.619$ OPT
- Computing such a PNE implies a 2.619-approx. algorithm
- In general, best response dynamics might need an exponential number of deviations before we arrive at a PNE

A coordination mechanism with potential function Φ and social cost function C is said to be β-nice if for any configuration s:

- $\Phi(s) \leq C(s)$
- $C(s) \leq \beta \mathrm{OPT} + 2\sum_i (c_i(s) - c_i(s_{-i}, s_i'))$

Lemma

The ProportionalSharing coordination mechanism is β-nice with $\beta = \frac{3+\sqrt{5}}{2} \approx 2.618$.

Approximation Algorithm

- We know that a PNE always exists for ProportionalSharing
- For any such PNE s we know that $\sum_i w_i c_i^{PS}(s) \leq 2.619$ OPT
- Computing such a PNE implies a 2.619-approx. algorithm
- In general, best response dynamics might need an exponential number of deviations before we arrive at a PNE

A coordination mechanism with potential function Φ and social cost function C is said to be β-nice if for any configuration s:

- $\Phi(s) \leq C(s)$
- $C(s) \leq \beta \mathrm{OPT} + 2\sum_i (c_i(s) - c_i(s_{-i}, s_i'))$

Lemma

The ProportionalSharing coordination mechanism is β-nice with $\beta = \frac{3+\sqrt{5}}{2} \approx 2.618$.

Approximation Algorithm

- We know that a PNE always exists for ProportionalSharing
- For any such PNE s we know that $\sum_i w_i c_i^{PS}(s) \leq 2.619$ OPT
- Computing such a PNE implies a 2.619-approx. algorithm
- In general, best response dynamics might need an exponential number of deviations before we arrive at a PNE

A coordination mechanism with potential function Φ and social cost function C is said to be β-nice if for any configuration s:

- $\Phi(s) \leq C(s)$
- $C(s) \leq \beta \mathrm{OPT} + 2\sum_i (c_i(s) - c_i(s_{-i}, s_i'))$

Lemma

The ProportionalSharing coordination mechanism is β-nice with $\beta = \frac{3+\sqrt{5}}{2} \approx 2.618$.

Approximation Algorithm

- We know that a PNE always exists for ProportionalSharing
- For any such PNE s we know that $\sum_i w_i c_i^{PS}(s) \leq 2.619$ OPT
- Computing such a PNE implies a 2.619-approx. algorithm
- In general, best response dynamics might need an exponential number of deviations before we arrive at a PNE

A coordination mechanism with potential function Φ and social cost function C is said to be β-nice if for any configuration s:

- $\Phi(s) \leq C(s)$
- $C(s) \leq \beta \mathrm{OPT} + 2\sum_i (c_i(s) - c_i(s_{-i}, s_i'))$

Lemma

The ProportionalSharing coordination mechanism is β-nice with $\beta = \frac{3+\sqrt{5}}{2} \approx 2.618$.

Approximation Algorithm

- We know that a PNE always exists for ProportionalSharing
- For any such PNE s we know that $\sum_i w_i c_i^{PS}(s) \leq 2.619$ OPT
- Computing such a PNE implies a 2.619-approx. algorithm
- In general, best response dynamics might need an exponential number of deviations before we arrive at a PNE

A coordination mechanism with potential function Φ and social cost function C is said to be β-nice if for any configuration s:

- $\Phi(s) \leq C(s)$
- $C(s) \leq \beta \text{OPT} + 2\sum_i (c_i(s) - c_i(s_{-i}, s_i'))$

Lemma

The ProportionalSharing coordination mechanism is β-nice with $\beta = \frac{3+\sqrt{5}}{2} \approx 2.618$.

Approximation Algorithm

- We know that a PNE always exists for ProportionalSharing
- For any such PNE s we know that $\sum_i w_i c_i^{PS}(s) \leq 2.619$ OPT
- Computing such a PNE implies a 2.619-approx. algorithm
- In general, best response dynamics might need an exponential number of deviations before we arrive at a PNE

A coordination mechanism with potential function Φ and social cost function C is said to be β-nice if for any configuration s:

- $\Phi(s) \leq C(s)$
- $C(s) \leq \beta \text{OPT} + 2\sum_i (c_i(s) - c_i(s_{-i}, s_i'))$

Lemma

The ProportionalSharing coordination mechanism is β-nice with $\beta = \frac{3+\sqrt{5}}{2} \approx 2.618$.

Approximation Algorithm

- We know that a PNE always exists for ProportionalSharing
- For any such PNE s we know that $\sum_i w_i c_i^{PS}(s) \leq 2.619$ OPT
- Computing such a PNE implies a 2.619-approx. algorithm
- In general, best response dynamics might need an exponential number of deviations before we arrive at a PNE

A coordination mechanism with potential function Φ and social cost function C is said to be β-nice if for any configuration s:

- $\Phi(s) \leq C(s)$
- $C(s) \leq \beta \mathrm{OPT} + 2\sum_i (c_i(s) - c_i(s_{-i}, s_i'))$

Lemma

The ProportionalSharing coordination mechanism is β-nice with $\beta = \frac{3+\sqrt{5}}{2} \approx 2.618$.

Approximation Algorithm

- We know that a PNE always exists for ProportionalSharing
- For any such PNE s we know that $\sum_i w_i c_i^{PS}(s) \leq 2.619$ OPT
- Computing such a PNE implies a 2.619-approx. algorithm
- In general, best response dynamics might need an exponential number of deviations before we arrive at a PNE

A coordination mechanism with potential function Φ and social cost function C is said to be β-nice if for any configuration s:

- $\Phi(s) \leq C(s)$
- $C(s) \leq \beta \mathrm{OPT} + 2\sum_i (c_i(s) - c_i(s_{-i}, s_i'))$

Lemma

The ProportionalSharing coordination mechanism is β-nice with $\beta = \frac{3+\sqrt{5}}{2} \approx 2.618$.

Approximation Algorithm

Corollary

Starting from some initial configuration s^0 and moving the player with the maximum absolute improvement in each step, leads to a profile s with $C^{PS}(s) \leq (2.619 + O(\epsilon))C^{SR}(s^*)$ in at most $O\left(\frac{n}{\epsilon} \log\left(\frac{\Phi(s^0)}{\Phi(s^*)}\right)\right)$ steps.

Conclusion

- SmithRule: Robust PoA is at most 4
- For any set of strongly local ordering policies the pure PoA is at least 4
- EqualSharing: Robust PoA is 2.5
- ProportionalSharing: Robust PoA is 2.619
- Combinatorial 2.619-approximation algorithm

Conclusion

- SmithRule: Robust PoA is at most 4
- For any set of strongly local ordering policies the pure PoA is at least 4
- EqualSharing: Robust PoA is 2.5
- ProportionalSharing: Robust PoA is 2.619
- Combinatorial 2.619-approximation algorithm

Conclusion

- SmithRule: Robust PoA is at most 4
- For any set of strongly local ordering policies the pure PoA is at least 4
- EqualSharing: Robust PoA is 2.5
- ProportionalSharing: Robust PoA is 2.619
- Combinatorial 2.619-approximation algorithm

Conclusion

- SmithRule: Robust PoA is at most 4
- For any set of strongly local ordering policies the pure PoA is at least 4
- EqualSharing: Robust PoA is 2.5
- ProportionalSharing: Robust PoA is 2.619
- Combinatorial 2.619-approximation algorithm

Conclusion

- SmithRule: Robust PoA is at most 4
- For any set of strongly local ordering policies the pure PoA is at least 4
- EqualSharing: Robust PoA is 2.5
- ProportionalSharing: Robust PoA is 2.619
- Combinatorial 2.619-approximation algorithm

Thank you!

Thank you!