Coordination Mechanisms

Dimitris Sakavalas

May 12, 2011

Outline

Introduction

Problems

Coordination models

Coordination Mechanisms(CM)

Selfish Scheduling

Congestion Games

Generalization of machine scheduling

Bounds for PoA of $(R||C_{max})$

Introduction

Objective

Create mechanisms to improve coordination of selfish agents

Idea: Modify players' objectives by introducing side payments
Examples: Selfish routing games (constant edge taxes),
Auctions (pay or penalize players to submit their true values)

Problems

Selfish Scheduling

m parallel links(machines), n selfish users. User i schedules load w_i on a machine j

PoA$=\Theta(\log m / \log\log m)$ (balls and bins)

Objective: Player i wants to minimize finishing time

Problems

Selfish Scheduling

m parallel links(machines), n selfish users. User i schedules load w_i on a machine j

PoA=$\Theta(\log m / \log \log m)$ (balls and bins)

Objective: Player i wants to minimize finishing time
Mechanism: Select scheduling policies of each machine
Conditions:Policies independent to the loads w_i(competitive analysis), Scheduling on a machine should depend only on the loads assigned to it (decentralized nature)

Problems

Selfish Scheduling

m parallel links(machines), n selfish users. User i schedules load w_i on a machine j

PoA$=\Theta(\log m / \log\log m)$ (balls and bins)

Objective: Player i wants to minimize finishing time
Mechanism: Select scheduling policies of each machine
Conditions:Policies independent to the loads w_i(competitive analysis), Scheduling on a machine should depend only on the loads assigned to it (decentralized nature)

$$PoA = \frac{\text{makespan of worst Nash equilibrium}}{\text{minimum makespan (independent of sceduling policies)}}$$

Problems

Congestion Games $(N, M, (\Sigma_i)_{i \in N}, (c^j)_{j \in M})$

N:set of players, M set of facilities(edges), Σ_i: collection of strategies for player i, $c^j : \mathbb{N} \rightarrow \mathbb{R}_+$: cost function of facility j

Problems

Congestion Games $(N, M, (\Sigma_i)_{i \in N}, (c^j)_{j \in M})$

N:set of players, M set of facilities(edges), Σ_i: collection of strategies for player i, $c^j : \mathbb{N} \to \mathbb{R}_+$: cost function of facility j

Generalization

- Players have loads $w = (w_1, \cdots, w_n)$
- Assymetric cost functions c_i^j. Cost of player i using facility j is $c_i^j(w^j)$ where w^j : sum of weights of the players using facility j

Problems

Congestion Games $(N, M, (\Sigma_i)_{i \in N}, (c^j)_{j \in M})$

N:set of players, M set of facilities(edges), Σ_i: collection of strategies for player i, $c^j : \mathbb{N} \to \mathbb{R}_+$: cost function of facility j

Generalization

- Players have loads $w = (w_1, \cdots, w_n)$
- Assymetric cost functions c_i^j. Cost of player i using facility j is $c_i^j(w^j)$ where w^j : sum of weights of the players using facility j

Mechanism

- Introduce delays: New cost functions $\hat{c}_i^j(w) \geq c_i^j(w)$
- Assign priorities to players: Facility j assigns priorities to players $t_1, t_2, \cdots$. Cost of t_k cannot be less than $c_{t_k}^j(w_{t_1} + \cdots + w_{t_k})$

Coordination models

Coordination models $(N, M, (\Sigma_i)_{i \in N}, (C^j)_{j \in M})$

N: set of players, M: set of facilities(edges), Σ_i: collection of strategies for player i, $A_i \in \Sigma_i$: set of facilities, $\mathbf{C^j}$ a **collection of cost functions** associated with j where $C^j \ni c^j : \mathbb{R}^n \to \mathbb{R}^n$.

Coordination models

Coordination models $(N, M, (\Sigma_i)_{i \in N}, (C^j)_{j \in M})$

N: set of players, M: set of facilities(edges), Σ_i: collection of strategies for player i, $A_i \in \Sigma_i$: set of facilities, **C^j** a **collection of cost functions** associated with j where $C^j \ni c^j : \mathbb{R}^n \to \mathbb{R}^n$.

Coordination model: Set of games

Coordination models

Coordination models $(N, M, (\Sigma_i)_{i \in N}, (C^j)_{j \in M})$

N: set of players, M: set of facilities(edges), Σ_i: collection of strategies for player i, $A_i \in \Sigma_i$: set of facilities, **C^j** a **collection of cost functions** associated with j where $C^j \ni c^j : \mathbb{R}^n \to \mathbb{R}^n$.

Coordination model: Set of games

Cost function

Players have load $w_1, \cdots, w_n$
When a player does not use the facility, his load is 0 and
$c_i^j(w_1, \cdots, w_{i-1}, \cdots, 0, w_{i+1}, \cdots w_n) = 0$

Coordination models

Coordination models $(N, M, (\Sigma_i)_{i \in N}, (C^j)_{j \in M})$

N: set of players, M: set of facilities(edges), Σ_i: collection of strategies for player i, $A_i \in \Sigma_i$: set of facilities, $\mathbf{C^j}$ a **collection of cost functions** associated with j where $C^j \ni c^j : \mathbb{R}^n \to \mathbb{R}^n$.

Coordination model: Set of games

Cost function

Players have load $w_1, \cdots, w_n$
When a player does not use the facility, his load is 0 and
$c_i^j(w_1, \cdots, w_{i-1}, \cdots, 0, w_{i+1}, \cdots w_n) = 0$

Symmetric game $G \in (N, M, (\Sigma_i)_{i \in N}, (C^j)_{j \in M})$

- $c_i^j = c_l^j \forall i, l$ players using j
- $c_i^j(w) = c^j \left(\sum_{i \text{ uses } j} w_i \right)$

Coordination models

Coordination model for selfish scheduling

$N = \{1, \cdots, n\}$: set of players, $M = \{1, \cdots, n\}$: set of machines/facilities, $\Sigma_i = \{\{1\}, \cdots, \{m\}\}$, c^j is a cost function if $\forall (w_1, ..., w_n)$ and $\forall S \subseteq N$, $\max_{i \in S} c_i^j(w_1, ..., w_n) \geq \sum_{i \in S} w_i$ (max finish time)

Facility j may introduce delay through c^j and order loads (cannot speed up execution)

Coordination models

Coordination model for selfish scheduling

$N = \{1, \cdots, n\}$: set of players, $M = \{1, \cdots, n\}$: set of machines/facilities, $\Sigma_i = \{\{1\}, \cdots, \{m\}\}$, c^j is a cost function if $\forall (w_1, ..., w_n)$ and $\forall S \subseteq N$, $\max_{i \in S} c_i^j(w_1, ..., w_n) \geq \sum_{i \in S} w_i$ (max finish time)

Facility j may introduce delay through c^j and order loads (cannot speed up execution)

Coordination model for weighted congestion game G

If $G : (N, M, (\Sigma_i)_{i \in N}, (c^j)_{j \in M})$, the coordination model for G is the set of all games G_i with cost functions $\hat{c}_i^j(w) \geq c_i^j(w), \forall j \in M, \forall w$

Coordination Mechanisms(CM)

Correspondence with competitive analysis

Coordination model $\leftrightarrow$ Online problem
Coordination mechanism $\leftrightarrow$ Online algorithm
Price of anarchy $\leftrightarrow$ Competitive ratio

Coordination Mechanisms(CM)

Correspondence with competitive analysis

Coordination model $\leftrightarrow$ Online problem
Coordination mechanism $\leftrightarrow$ Online algorithm
Price of anarchy $\leftrightarrow$ Competitive ratio

Definition

A **coordination mechanism** for a coordination model $(N, M, (\Sigma_i)_{i \in N}, (C^j)_{j \in M})$ is a set of cost functions, one for each facility

Coordination Mechanisms(CM)

Correspondence with competitive analysis

Coordination model $\leftrightarrow$ Online problem
Coordination mechanism $\leftrightarrow$ Online algorithm
Price of anarchy $\leftrightarrow$ Competitive ratio

Definition

A **coordination mechanism** for a coordination model $(N, M, (\Sigma_i)_{i \in N}, (C^j)_{j \in M})$ is a set of cost functions, one for each facility

Social cost(sc)-Social optimum(opt)

CM $c = (c^1, ..., c^m)$, set of loads $w = (w_1, ..., w_n)$, set of strategies $A = (A_1, ..., A_n) \in \Sigma_1, ..., \Sigma_n$, $cost_i$:cost incurred by player i

- $sc(w; c; A) = \max_{i \in N} cost_i$
- $opt(w) = \inf_{c, A} sc(w; c; A)$ (independent of the CMs)

Price of Anarchy

To CM c and w corresponds a game G
Ne(**w**; **c**): the set of (mixed) Nash equilibria of G

PoA or Coordination ratio of a CM c

$$PA(c) = \sup_{w} \sup_{E \in Ne(w;c)} \frac{sc(w; c; E)}{opt(w)}$$

Price of Anarchy

To CM c and w corresponds a game G
Ne(**w**; **c**): the set of (mixed) Nash equilibria of G

PoA or Coordination ratio of a CM c

$$PA(c) = \sup_{w} \sup_{E \in Ne(w;c)} \frac{sc(w;c;E)}{opt(w)}$$

PoA of a coordination model

The minimum PoA over all its CMs.

Selfish Scheduling

Hint: Approximation algorithm (greedy) of approximation ratio $4/3 - 1/3m$. Loads ordered in decreasing size (LPT scheduling)

Selfish Scheduling

Hint: Approximation algorithm (greedy) of approximation ratio $4/3 - 1/3m$. Loads ordered in decreasing size (LPT scheduling)

Observation

Symmetric CM with the same scheduling policy on each facility have large PoA due to existence of NE where players select facilities uniformly at random (players "collide")

Example: $n = m$ players with load 1

Selfish Scheduling

Hint: Approximation algorithm (greedy) of approximation ratio $4/3 - 1/3m$. Loads ordered in decreasing size (LPT scheduling)

Observation

Symmetric CM with the same scheduling policy on each facility have large PoA due to existence of NE where players select facilities uniformly at random (players "collide")
Example: $n = m$ players with load 1

Situation avoided in pure equilibria
Idea: Introduce delays to break "symmetry"

Selfish Scheduling

Hint: Approximation algorithm (greedy) of approximation ratio $4/3 - 1/3m$. Loads ordered in decreasing size (LPT scheduling)

Observation

Symmetric CM with the same scheduling policy on each facility have large PoA due to existence of NE where players select facilities uniformly at random (players "collide")
Example: $n = m$ players with load 1

Situation avoided in pure equilibria
Idea: Introduce delays to break "symmetry"

Coordination Mechanism 1

- Each machine scedules jobs in decreasing order
- machine j intodoces delay $j\varepsilon$ for small $\varepsilon > 0$

Selfish Scheduling

Hint: Approximation algorithm (greedy) of approximation ratio $4/3 - 1/3m$. Loads ordered in decreasing size (LPT scheduling)

Observation

Symmetric CM with the same scheduling policy on each facility have large PoA due to existence of NE where players select facilities uniformly at random (players "collide")
Example: $n = m$ players with load 1

Situation avoided in pure equilibria
Idea: Introduce delays to break "symmetry"

Coordination Mechanism 1

- Each machine scedules jobs in decreasing order
- machine j intodeuces delay $j\varepsilon$ for small $\varepsilon > 0$

Drawback: Jobs not of distinct size, delays $j\varepsilon$ create ties.

Selfish Scheduling

Coordination mechanism $\mathcal{C}$ for selfish scheduling

1. Each machine scedules jobs in decreasing order, lexicographic order to break ties (based on jobs' ID)
2. Different cost on the facilities for each player (unique optimal machine for each player)

Selfish Scheduling

Coordination mechanism $\mathcal{C}$ for selfish scheduling

1. Each machine scedules jobs in decreasing order, lexicographic order to break ties (based on jobs' ID)
2. Different cost on the facilities for each player (unique optimal machine for each player)

Cost function

Let $\delta > 0$, suppose that job i is to finish at time t_i in machine j. The machine will release i at time t'

$$t' = c_i^j(w_1, ..., w_n) \text{ where } t' = \min_{k \in [t,(1+\delta)t]} \{k : k \mod m+1 = j\}$$

(representation of t' in the $(m+1) - ary$ system ends in j)

Selfish Scheduling

Coordination mechanism $\mathcal{C}$ for selfish scheduling

1. Each machine scedules jobs in decreasing order, lexicographic order to break ties (based on jobs' ID)
2. Different cost on the facilities for each player (unique optimal machine for each player)

Cost function

Let $\delta > 0$, suppose that job i is to finish at time t_i in machine j. The machine will release i at time t'

$$t' = c_i^j(w_1, \ldots, w_n) \text{ where } t' = \min_{k \in [t,(1+\delta)t]} \{k : k \mod m+1 = j\}$$

(representation of t' in the $(m+1)-ary$ system ends in j)

In $\mathcal{C}$ with the above cost function, there is a unique minimum cost facility for each player

Selfish Scheduling

Theorem

Coordination mechanism $\mathcal{C}$ *for* n *players and* m *machines has PoA* $4/3 - 1/(3m)$

Selfish Scheduling

Theorem

Coordination mechanism C for n players and m machines has PoA $4/3 - 1/(3m)$

Proof.

For the $i - th$ greater load there is a unique facility with minimum cost independently of the smaller loads. Exactly the LPT scheduling , approximation ratio $4/3 - 1/(3m)$.

Delay introduced by δ increases social cost by at most δ

$$PoA = inf_{\delta}(4/3 - 1/(3m) + \delta) = 4/3 - 1/(3m) \quad \square$$

Selfish Scheduling

Theorem

Coordination mechanism $\mathcal{C}$ for n players and m machines has PoA $4/3 - 1/(3m)$

Proof.

For the $i - th$ greater load there is a unique facility with minimum cost independently of the smaller loads. Exactly the LPT scheduling , approximation ratio $4/3 - 1/(3m)$.

Delay introduced by δ increases social cost by at most δ

$$PoA = inf_{\delta}(4/3 - 1/(3m) + \delta) = 4/3 - 1/(3m) \quad \square$$

Observation

- There is a unique NE and it has low computational complexity
- Compute $PoA(\mathcal{C}) \leq_p$ Compute approximation ratio of LPT

Selfish Scheduling

Theorem

Coordination mechanism $\mathcal{C}$ for n players and m machines has PoA $4/3 - 1/(3m)$

Proof.

For the $i-th$ greater load there is a unique facility with minimum cost independently of the smaller loads. Exactly the LPT scheduling , approximation ratio $4/3 - 1/(3m)$.
Delay introduced by δ increases social cost by at most δ

$$PoA = inf_{\delta}(4/3 - 1/(3m) + \delta) = 4/3 - 1/(3m) \quad \square$$

Observation

- There is a unique NE and it has low computational complexity
- Compute $PoA(\mathcal{C}) \leq_p$ Compute approximation ratio of LPT

Theorem

$\mathcal{C}$ for n players and m machines with different speeds has PoA $2 - 2/(m+1)$

Congestion Games

Proposition

Without a CM, the PoA of congestion games is unbounded

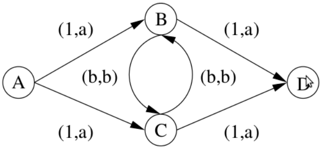
Congestion Games

Proposition

Without a CM, the PoA of congestion games is unbounded

Proof(Example)

Single-commodity game with $n = 2$ players and $a \gg b \gg 1$

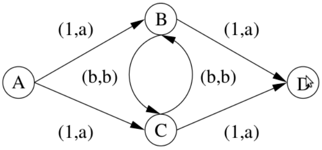


Congestion Games

Proposition

Without a CM, the PoA of congestion games is unbounded

Proof(Example)
Single-commodity game with $n = 2$ players and $a \gg b \gg 1$



NE: $A = (A_1, A_2) \in \Sigma_1 \times \Sigma_2$
$(A_1, A_2) = (ABCD, ACBD)$
OPT:$A' = (A'_1, A'_2) = (ABD, ACD)$

$PoA = sc(A)/sc(a') = (2 + b)/2$
arbitarily high

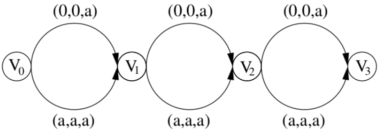
□

Series Parallel Congestion Games

Theorem

There are congestion games (even series parallel) for which no CM has PoA $\geq n$, n number of players

Proof(Example)
Network with nodes: $v_0, \ldots, v_n$, parallel edges: $(v_i, v_i + 1)$, upper edge costs: $(0, \ldots, 0, a)$, lower edge costs: $(a, \ldots, a)$

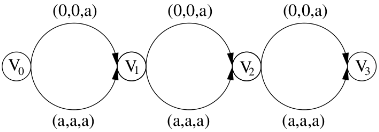


Series Parallel Congestion Games

Theorem

There are congestion games (even series parallel) for which no CM has PoA $\geq n$, n number of players

Proof(Example)
Network with nodes: $v_0, \ldots, v_n$, parallel edges: $(v_i, v_i + 1)$, upper edge costs: $(0, \ldots, 0, a)$, lower edge costs: $(a, \ldots, a)$



NE:$A \rightarrow$ All players select upper edges
OPT:$A' \rightarrow$ Player i selects upper edges except between u_{i-1}, u_i

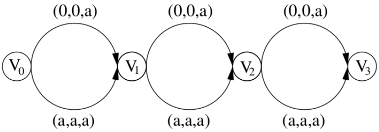
$$PoA = sc(A)/sc(A') = n \cdot a/n = n$$

Series Parallel Congestion Games

Theorem

There are congestion games (even series parallel) for which no CM has $PoA \geq n$, n *number of players*

Proof(Example)
Network with nodes: $v_0, \ldots, v_n$, parallel edges: $(v_i, v_i + 1)$, upper edge costs: $(0, \ldots, 0, a)$, lower edge costs: $(a, \ldots, a)$



NE:$A \rightarrow$ All players select upper edges
OPT:$A' \rightarrow$ Player i selects upper edges except between u_{i-1}, u_i

$$PoA = sc(A)/sc(A') = n \cdot a/n = n$$

Symmetric CM $\mathcal{C}$. In $\mathcal{C}$ at least one player incurs cost at least a between u_{i-1}, u_i All stages are independent, so $\exists NE$ s.t. the same player incurs cost at least a in every stage. $PoA = n$ □

Series Parallel Congestion Games

Theorem

For every series-parallel congestion game there is a CM with $PoA \leq n$

Series Parallel Congestion Games

Theorem

For every series-parallel congestion game there is a CM with $PoA \leq n$

Potential $P(A)$

$A = (A_1, ..., A_n)$: set of strategies, n^e: number of occurences of edge e in the paths $A_1, ..., A_n$ then $P(A) = \sum_e \sum_{k=1}^{n^e} c^e(k)$

A is a $NE \Leftrightarrow P(A)$: local minimum

Series Parallel Congestion Games

Theorem

For every series-parallel congestion game there is a CM with $PoA \leq n$

Potential $P(A)$

$A = (A_1, ..., A_n)$: set of strategies, n^e: number of occurences of edge e in the paths $A_1, ..., A_n$ then $P(A) = \sum_e \sum_{k=1}^{n^e} c^e(k)$

A is a $NE \Leftrightarrow P(A)$: local minimum

Lemma 1

$\forall A : sc(A) \leq P(A) \leq n \cdot sc(A)$

Proof

- $sc(A) = \max_i c_i = max_i \sum_{e \in A_i} c^e(n^e) \leq \sum_e c^e(n^e) \leq$
  $\leq \sum_e \sum_{k=1}^{n^e} c^e(k) = P(A)$
- $P(A) = \sum_e \sum_{k=1}^{n^e} c^e(k) \leq \sum_e n^e c^e(n^e) = \sum_i c_i \leq$
  $\leq n \max_i c_i = n \cdot sc(A)$

CM for series-parallel networks

Coordination Mechanism $\mathcal{C}$

Let $A^* = (A_1^*, ..., A_2^*)$ an optimal set of strategies, large $a \gg 1$

$$\hat{c}^e(k) = \begin{cases} c^e(k), k \leq n^e(A^*) \\ a \cdot m, \forall k \text{ when } n^e(A^*) = 0 \\ a, \quad \text{otherwise} \end{cases}$$

CM for series-parallel networks

Coordination Mechanism $\mathcal{C}$

Let $A^* = (A^*_1, ..., A^*_2)$ an optimal set of strategies, large $a \gg 1$

$$\hat{c}^e(k) = \begin{cases} c^e(k), k \leq n^e(A^*) \\ a \cdot m, \forall k \text{ when } n^e(A^*) = 0 \\ a, \qquad \text{otherwise} \end{cases}$$

High cost a will discourage players to use edge e more than $n^e(A^*)$ times. $P(A) = P(A^*)$??

CM for series-parallel networks

Coordination Mechanism $\mathcal{C}$

Let $A^* = (A_1^*, ..., A_2^*)$ an optimal set of strategies, large $a \gg 1$

$$\hat{c}^e(k) = \begin{cases} c^e(k), k \leq n^e(A^*) \\ a \cdot m, \forall k \text{ when } n^e(A^*) = 0 \\ a, \quad \text{otherwise} \end{cases}$$

High cost a will discourage players to use edge e more than $n^e(A^*)$ times. $P(A) = P(A^*)$??

Lemma 2

$A_1^*, ..., A_n^*$: edge-disjoint $s - t$ paths in a series-parallel multi-graph, $A_1, ..., A_k$: any other $s - t$ paths with $k < n$. Then $\exists\ s - t$ path with edges that appear in $A_1^*, ..., A_n^*$ but not in $A_1, ..., A_k$

CM for series-parallel networks

Proof of Theorem

Series parallel(directed) graph, optimalset of strategies $A^* = (A_1^*, ..., A_n^*)$, NE $A = (A_1, ..., A_n)$.

We will show that $\forall e, n^e(A) \leq n^e(A^*)$

CM for series-parallel networks

Proof of Theorem

Series parallel(directed) graph, optimalset of strategies $A^* = (A_1^*, ..., A_n^*)$, NE $A = (A_1, ..., A_n)$.

We will show that $\forall e, n^e(A) \leq n^e(A^*)$

- Paths in A use only edges that appear in A^*, else i would switch to any low cost path in A^*

CM for series-parallel networks

Proof of Theorem

Series parallel(directed) graph, optimalset of strategies
$A^* = (A_1^*, ..., A_n^*)$, NE $A = (A_1, ..., A_n)$.
We will show that $\forall e, n^e(A) \leq n^e(A^*)$

- Paths in A use only edges that appear in A^*, else i would switch to any low cost path in A^*
- Arbitary player i, A_{-i} paths of remaining players.
  Lemma 2 $\Rightarrow \exists$ path p s.t. $n^e(A_{-i}) \leq n^e(A^*) - 1, \forall e \in p$

CM for series-parallel networks

Proof of Theorem

Series parallel(directed) graph, optimalset of strategies
$A^* = (A_1^*, ..., A_n^*)$, NE $A = (A_1, ..., A_n)$.
We will show that $\forall e, n^e(A) \leq n^e(A^*)$

- Paths in A use only edges that appear in A^*, else i would switch to any low cost path in A^*
- Arbitary player i, A_{-i} paths of remaining players.
  Lemma 2 $\Rightarrow \exists$ path p s.t. $n^e(A_{-i}) \leq n^e(A^*) - 1, \forall e \in p$

  If i uses edge e' with $n^{e'}(A^*) \leq n^{e'}(A)$ then p is a strategy for player i with $n^e(A_i) \leq n^e(A^*), \forall e \in p$ (less expensive)

  A is NE $\Rightarrow$ Player i only uses edges e with $n^e(A) \leq n^e(A^*)$

CM for series-parallel networks

Proof of Theorem

Series parallel(directed) graph, optimalset of strategies
$A^* = (A_1^*, ..., A_n^*)$, NE $A = (A_1, ..., A_n)$.
We will show that $\forall e, n^e(A) \leq n^e(A^*)$

- Paths in A use only edges that appear in A^*, else i would switch to any low cost path in A^*
- Arbitary player i, A_{-i} paths of remaining players.
  Lemma 2 $\Rightarrow \exists$ path p s.t. $n^e(A_{-i}) \leq n^e(A^*) - 1, \forall e \in p$

  If i uses edge e' with $n^{e'}(A^*) \leq n^{e'}(A)$ then p is a strategy for player i with $n^e(A_i) \leq n^e(A^*), \forall e \in p$ (less expensive)

  A is NE $\Rightarrow$ Player i only uses edges e with $n^e(A) \leq n^e(A^*)$

Hence $P(A) \leq P(A^*)$ and Lemma 1$\Rightarrow sc(A) \leq n \cdot sc(A^*) \Rightarrow$
$PoA = \sup_A \frac{sc(A)}{sc(A^*)} \leq n$ □

Generalization of machine scheduling

Unrelated machine sceduling ($R||C_{max}$)

n players/jobs, m machines , p_{ij} processing time of job i in machine j, μ schedule function: maps each job to a machine, $M_j = \sum_{i:j=\mu(i)} p_{ij}$ makespan of machine j.

Different assumptions on p_{ij} yield different sceduling problems

Generalization of machine scheduling

Unrelated machine sceduling ($R||C_{max}$)

n players/jobs, m machines , p_{ij} processing time of job i in machine j, μ schedule function: maps each job to a machine, $M_j = \sum_{i:j=\mu(i)} p_{ij}$ makespan of machine j.

Different assumptions on p_{ij} yield different sceduling problems

Identical Machine Scheduling ($P||C_{max}$)

For each job i and machines j, k $p_{ij} = p_{ik} = p_i$

Generalization of machine scheduling

Unrelated machine sceduling ($R||C_{max}$)

n players/jobs, m machines , p_{ij} processing time of job i in machine j, μ schedule function: maps each job to a machine, $M_j = \sum_{i:j=\mu(i)} p_{ij}$ makespan of machine j.

Different assumptions on p_{ij} yield different sceduling problems

Identical Machine Scheduling ($P||C_{max}$)

For each job i and machines j, k $p_{ij} = p_{ik} = p_i$

Uniform/related machine sceduling ($Q||C_{max}$)

$p_{ij} = p_i/s_j$ where p_i processing requirement of job i and s_j speed of machine j.

Generalization of machine scheduling

Unrelated machine sceduling ($R||C_{max}$)

n players/jobs, m machines , p_{ij} processing time of job i in machine j, μ schedule function: maps each job to a machine, $M_j = \sum_{i:j=\mu(i)} p_{ij}$ makespan of machine j.

Different assumptions on p_{ij} yield different sceduling problems

Identical Machine Scheduling ($P||C_{max}$)

For each job i and machines j, k $p_{ij} = p_{ik} = p_i$

Uniform/related machine sceduling ($Q||C_{max}$)

$p_{ij} = p_i/s_j$ where p_i processing requirement of job i and s_j speed of machine j.

Restricted assignment or bipartite sceduling ($B||C_{max}$)

Job i can be scheduled on $S_i \subseteq M$. $p_{ij} = p_i$, if $j \in S_i$ and $p_{ij} = \infty$ otherwise

Different Coordination Mechanisms

Coordination Mechanisms (sets of sceduling policies)

- ShortestFirst: non-decreasing order of jobs
- LongestFirst: non-increasing order of jobs
- Randomized: random order of jobs
- Makespan: Process all jobs on the same machine in parallel ($p_{ji} = M_j$)

Different Coordination Mechanisms

Coordination Mechanisms (sets of sceduling policies)

- ShortestFirst: non-decreasing order of jobs
- LongestFirst: non-increasing order of jobs
- Randomized: random order of jobs
- Makespan: Process all jobs on the same machine in parallel ($p_{ji} = M_j$)

Price of anarchy for the different policies and scheduling problems

	Makespan	ShortestFirst	LongestFirst	Randomized
$P \| \|C_{\max}$	$2 - \frac{2}{m+1}$	$2 - \frac{1}{m}$	$\frac{4}{3} - \frac{1}{3m}$	$2 - \frac{2}{m+1}$
$Q \| \|C_{\max}$	$\Theta(\frac{\log m}{\log \log m})$	$\Theta(\log m)$	$1.52 \leq P \leq 1.59$	$\Theta(\frac{\log m}{\log \log m})$
$B \| \|C_{\max}$	$\Theta(\frac{\log m}{\log \log m})$	$\Theta(\log m)$	$\Theta(\log m)$	$\Theta(\frac{\log m}{\log \log m})$
$R \| \|C_{\max}$	Unbounded	$\log m \leq P \leq m$	Unbounded	$\Theta(m)$

Scheduling Policies

Policy is run locally at each machine, no access to information for the global state (P_j policy for machine j , S_j jobs assigned to j)

Scheduling Policies

Policy is run locally at each machine, no access to information for the global state (P_j policy for machine j , S_j jobs assigned to j)

- **Strong local policy** P_j**:** Only makes use of processing time of jobs $i \in S_j$ on j and assigns i a completition time $P_j(S_j, i)$
- **Local policy** P_j**:** Makes use of all parameters of jobs $i \in S_j$ and assigns each i a completition time $P_j(S_j, i)$ (Ex. Uses processing times of $i \in S_j$ on other machines)

Scheduling Policies

Policy is run locally at each machine, no access to information for the global state (P_j policy for machine j , S_j jobs assigned to j)

- **Strong local policy P_j:** Only makes use of processing time of jobs $i \in S_j$ on j and assigns i a completition time $P_j(S_j, i)$
- **Local policy P_j:** Makes use of all parameters of jobs $i \in S_j$ and assigns each i a completition time $P_j(S_j, i)$ (Ex. Uses processing times of $i \in S_j$ on other machines)
- **Non-preemtive policy:** Processes each job in an uninterrupted fashion without any delay
- **Independence of irrelevant alternatives property(IIA):** For any set S of jobs and $i, i' \in S$, then $\forall k$ job $P_j(S, i) < P_j(S, i') \Rightarrow P_j(S \cup \{k\}, i) < P_j(S \cup \{k\}, i')$,
- **Ordering policy**: Orders the jobs non-preemptively based on a global ordering (deterministic non-preemtive policy with IIA is an ordering policy)

Upper bound for PoA of $(R||C_{max})$

Notation

- $p_i = \min_j p_{ij}$
- **Inefficiency** of job i: $e_{ij} = p_{ij}/p_i$
- **min-weight** of set S: $\sum_{i \in S} p_i$
- $W = \sum_{1 \leq i \leq n} p_i$
- M_{kj}: jobs(parts) processed on j after time $2kOPT$ in a PNE
- $M_k = \bigcup_{1 \leq j \leq m} M_{kj}$
- $R_{kj} = \sum_{i \in M_{kj}} p_i$, if job i partially processed on j for x units of time after $2kOPT$, contributes $x/e_{ij} = xp_i/p_{ij}$ to R_{kj}

Upper bound for PoA of $(R||C_{max})$

Notation

- $p_i = \min_j p_{ij}$
- **Inefficiency** of job i: $e_{ij} = p_{ij}/p_i$
- **min-weight** of set S: $\sum_{i \in S} p_i$
- $W = \sum_{1 \le i \le n} p_i$
- M_{kj}: jobs(parts) processed on j after time $2kOPT$ in a PNE
- $M_k = \bigcup_{1 \le j \le m} M_{kj}$
- $R_{kj} = \sum_{i \in M_{kj}} p_i$, if job i partially processed on j for x units of time after $2kOPT$, contributes $x/e_{ij} = xp_i/p_{ij}$ to R_{kj}

Inefficiency-based policy

Each machine j orders the jobs assigned to it in the non-decreasing order of their inefficiency e_{ij}

Upper bound for PoA of $(R||C_{max})$

Theorem

PoA for $(R||C_{max})$ for the inefficiency-based policy is at most $2 \log m + 4$

Upper bound for PoA of $(R||C_{max})$

Theorem

PoA for $(R||C_{max})$ *for the inefficiency-based policy is at most* $2 \log m + 4$

Lemma

$\forall k \geq 1, R_k \leq \frac{1}{2} \cdot R_{k-1}$

Upper bound for PoA of $(R||C_{max})$

Theorem

PoA for $(R||C_{max})$ for the inefficiency-based policy is at most $2\log m + 4$

Lemma

$\forall k \geq 1, R_k \leq \frac{1}{2} \cdot R_{k-1}$

Proof(Lemma).
O_j jobs processed on machine j by OPT, $O_{kj} = O_j \cap M_k$,
f_{kj} minimum inefficiency (on machine j) of all $i \in O_{kj}$ in the NE assignment**.

Upper bound for PoA of $(R||C_{max})$

Theorem

PoA for $(R||C_{max})$ for the inefficiency-based policy is at most $2\log m + 4$

Lemma

$\forall k \geq 1, R_k \leq \frac{1}{2} \cdot R_{k-1}$

Proof(Lemma).
O_j jobs processed on machine j by OPT, $O_{kj} = O_j \cap M_k$,
f_{kj} minimum inefficiency (on machine j) of all $i \in O_{kj}$ in the NE assignment**.
If $O_{kj} \neq \emptyset$ then in the NE assignment all jobs i on j with $e_{ij} \leq f_{kj}$ have $ct(i) \leq (2k-1)OPT$,
Otherwise $i \in O_{kj}$ with $e_{ij} = f_{kj}$ would move to j and have $ct(i) \leq (2k-1)OPT + OPT = 2kOPT$.

Proof (cont'd)

Hence j processes jobs i of $e_{ij} \leq f_{kj}$ between times $(2k-2)OPT$ and $(2k-1)OPT$ which implies

$$R_{k-1,j} - R_{kj} \geq OPT/f_{kj} \tag{1}$$

Proof (cont'd)

Hence j processes jobs i of $e_{ij} \leq f_{kj}$ between times $(2k-2)OPT$ and $(2k-1)OPT$ which implies

$$R_{k-1,j} - R_{kj} \geq OPT/f_{kj} \tag{1}$$

But $i \in O_{kj}$ processed by OPT on j with inefficiency $e_{ij} \geq f_{kj}$ hence min-weight of O_{kj} is

$$\sum_{i \in O_{kj}} p_i \leq \frac{\sum_{i \in O_{kj}} p_{ij}}{f_{kj}} = OPT/f_{kj} \tag{2}$$

Proof (cont'd)

Hence j processes jobs i of $e_{ij} \leq f_{kj}$ between times $(2k-2)OPT$ and $(2k-1)OPT$ which implies

$$R_{k-1,j} - R_{kj} \geq OPT/f_{kj} \tag{1}$$

But $i \in O_{kj}$ processed by OPT on j with inefficiency $e_{ij} \geq f_{kj}$ hence min-weight of O_{kj} is

$$\sum_{i \in O_{kj}} p_i \leq \frac{\sum_{i \in O_{kj}} p_{ij}}{f_{kj}} = OPT/f_{kj} \tag{2}$$

$(1),(2) \Rightarrow R_{k-1,j} - R_{kj} \geq$min weight of $i \in O_{kj}$.
Sum over all j, since $M_k = \cup_j O_{kj}$

$$R_{k-1} - R_k \geq R_k \Rightarrow R_{k-1} \geq 2R_k \quad \square$$

Proof(Theorem)

For $k = b = \lceil \log m \rceil$
$Lemma \Rightarrow R_b \leq \frac{R_{b-1}}{2} = \frac{R_{b-2}}{4} = \cdots = \frac{R_0}{m} = \frac{W}{m} \leq OPT$
(Total processing time of jobs i with $e_{ij} = 1$ at most OPT)

Hence $\forall i$ job , $ct(i) \leq 2bOPT + OPT = (2b+1)OPT$
(worst-case, all jobs with $ct(i) > 2bOPT$ move to the same machine)

Proof(Theorem)

For $k = b = \lceil \log m \rceil$
$Lemma \Rightarrow R_b \leq \frac{R_{b-1}}{2} = \frac{R_{b-2}}{4} = \cdots = \frac{R_0}{m} = \frac{W}{m} \leq OPT$
(Total processing time of jobs i with $e_{ij} = 1$ at most OPT)

Hence $\forall i$ job , $ct(i) \leq 2bOPT + OPT = (2b+1)OPT$
(worst-case, all jobs with $ct(i) > 2bOPT$ move to the same machine)

Let job i with $ct(i) > 2bOPT$, i moves on j with $e_{ij} = 1$
Start time $\leq (2b+1)OPT$, $ct(i) \leq (2b+2)OPT$

Proof(Theorem)

For $k = b = \lceil \log m \rceil$
$Lemma \Rightarrow R_b \leq \frac{R_{b-1}}{2} = \frac{R_{b-2}}{4} = \cdots = \frac{R_0}{m} = \frac{W}{m} \leq OPT$
(Total processing time of jobs i with $e_{ij} = 1$ at most OPT)

Hence $\forall i$ job , $ct(i) \leq 2bOPT + OPT = (2b+1)OPT$
(worst-case, all jobs with $ct(i) > 2bOPT$ move to the same machine)

Let job i with $ct(i) > 2bOPT$, i moves on j with $e_{ij} = 1$
Start time $\leq (2b+1)OPT$, $ct(i) \leq (2b+2)OPT$

Since assignment is a NE
$max_i ct(i) \leq (2b+2)OPT \leq (2\log m + 4)OPT \quad \square$