Efficiency of Nash Equilibria

Fall 2020

Efficiency at equilibrium

- We have analyzed the existence of PNE and NE.
- The players' goals can be different from those of the society.
- Fixing a social goal, then an optimal situation can be defined.
- How good/bad are NE with respect to this goal?

Efficiency at equilibrium

- We have analyzed the existence of PNE and NE.
- The players' goals can be different from those of the society.
- Fixing a social goal, then an optimal situation can be defined.
- How good/bad are NE with respect to this goal?
- How far are NE from the optimal social goal?

Efficiency at equilibrium

- We have analyzed the existence of PNE and NE.
- The players' goals can be different from those of the society.
- Fixing a social goal, then an optimal situation can be defined.
- How good/bad are NE with respect to this goal?
- How far are NE from the optimal social goal?
- To perform such analysis for strategic games, first we have to define a social function to optimize, this function is usually called the social cost or social utility.

Social cost

Consider a n-player game $\Gamma = (A_1, \ldots, A_n, c_1, \ldots, c_n)$. Let

- $A = A_1 \times \cdots \times A_n$,
- $PNE(\Gamma)$ be the set of PNE of Γ,
- $NE(\Gamma)$ be the set of NE of Γ,

Social cost

Consider a n-player game $\Gamma = (A_1, \ldots, A_n, c_1, \ldots, c_n)$. Let

- $A = A_1 \times \cdots \times A_n$,
- $PNE(\Gamma)$ be the set of PNE of Γ,
- $NE(\Gamma)$ be the set of NE of Γ,
- $\mathcal{C} : A \to \mathbb{R}$ be a social cost function.

  $\mathcal{C}$ can be extended to mixed strategy profiles by computing the average under the joint product distribution.

Usual social cost functions

Usual social cost functions

- Utilitarian social cost : $C(s) = \sum_{i \in N} c_i(s)$.

Usual social cost functions

- Utilitarian social cost : $C(s) = \sum_{i \in N} c_i(s)$.
- Egalitarian social cost: $C(s) = \max_{i \in N} c_i(s)$.

Usual social cost functions

- Utilitarian social cost : $C(s) = \sum_{i \in N} c_i(s)$.
- Egalitarian social cost: $C(s) = \max_{i \in N} c_i(s)$.
- Game specific cost/utility defined by the model motivating the game.

Price of Anarchy/Stability

Price of Anarchy/Stability

The Price of anarchy of Γ is defined as

$$PoA(\Gamma) = \frac{\max_{\sigma \in NE(\Gamma)} C(\sigma)}{\min_{s \in A} C(s)}.$$

Price of Anarchy/Stability

The Price of anarchy of Γ is defined as

$$PoA(\Gamma) = \frac{\max_{\sigma \in NE(\Gamma)} C(\sigma)}{\min_{s \in A} C(s)}.$$

The Price of stability of Γ is defined as

$$PoS(\Gamma) = \frac{\min_{\sigma \in NE(\Gamma)} C(\sigma)}{\min_{s \in A} C(s)}.$$

Price of Anarchy/Stability

The Price of anarchy of Γ is defined as

$$PoA(\Gamma) = \frac{\max_{\sigma \in NE(\Gamma)} C(\sigma)}{\min_{s \in A} C(s)}.$$

The Price of stability of Γ is defined as

$$PoS(\Gamma) = \frac{\min_{\sigma \in NE(\Gamma)} C(\sigma)}{\min_{s \in A} C(s)}.$$

For social utility functions the terms are inverted in the definition.

Price of Anarchy/Stability

Price of Anarchy/Stability

- For games having a PNE, we might be interested in those values over $PNE(\Gamma)$ instead of $NE(\Gamma)$.

Price of Anarchy/Stability

- For games having a PNE, we might be interested in those values over $PNE(\Gamma)$ instead of $NE(\Gamma)$.
- For families of games, we might be interested in analyzing PoA and PoS as a function of some parameter. For example the number of players.

Price of Anarchy/Stability

- For games having a PNE, we might be interested in those values over $PNE(\Gamma)$ instead of $NE(\Gamma)$.
- For families of games, we might be interested in analyzing PoA and PoS as a function of some parameter. For example the number of players.

- PoA measures the worst decentralized equilibrium scenario giving the maximum system degradation.

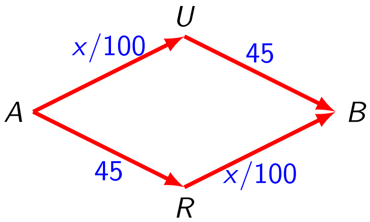
Price of Anarchy/Stability

- For games having a PNE, we might be interested in those values over $PNE(\Gamma)$ instead of $NE(\Gamma)$.
- For families of games, we might be interested in analyzing PoA and PoS as a function of some parameter. For example the number of players.

- PoA measures the worst decentralized equilibrium scenario giving the maximum system degradation.
- PoS measures the best decentralized equilibrium scenario giving the best possible degradation.

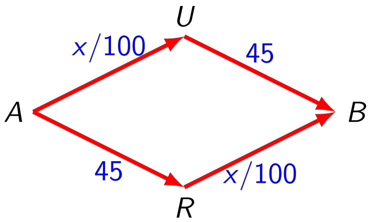
Network

- 4000 drivers drive from A to B on



Network

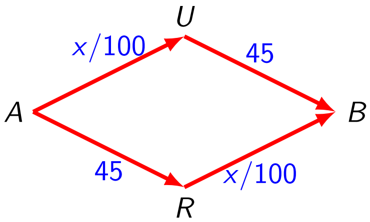
- 4000 drivers drive from A to B on



- Set the social cost to be the maximum travel time.

Network

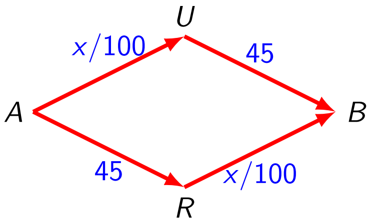
- 4000 drivers drive from A to B on



- Set the social cost to be the maximum travel time.
- Optimal social cost is reached when half of the drivers take $A - U - B$ and the other half $A - R - B$ with social cost 65.

Network

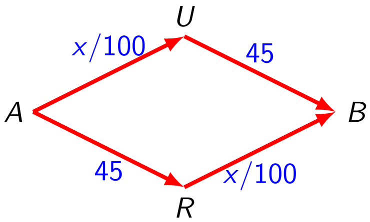
- 4000 drivers drive from A to B on



- Set the social cost to be the maximum travel time.
- Optimal social cost is reached when half of the drivers take $A - U - B$ and the other half $A - R - B$ with social cost 65.
- In the NE

Network

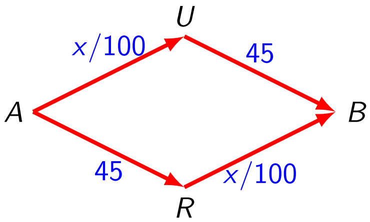
- 4000 drivers drive from A to B on



- Set the social cost to be the maximum travel time.
- Optimal social cost is reached when half of the drivers take $A - U - B$ and the other half $A - R - B$ with social cost 65.
- In the NE the same configuration.

Network

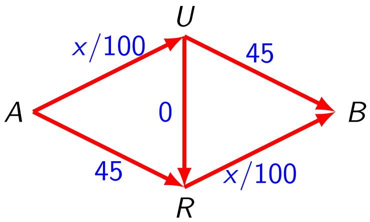
- 4000 drivers drive from A to B on



- Set the social cost to be the maximum travel time.
- Optimal social cost is reached when half of the drivers take $A - U - B$ and the other half $A - R - B$ with social cost 65.
- In the NE the same configuration.
- $PoA = PoS = 65/65 = 1$

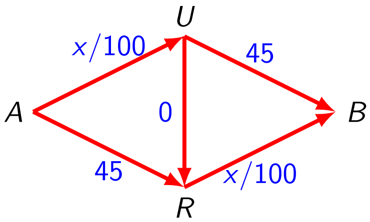
Braess' Network

- 4000 drivers drive from A to B on



Braess' Network

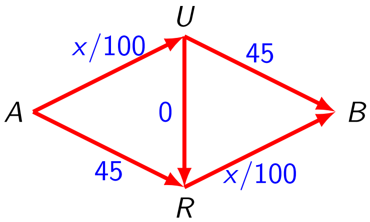
- 4000 drivers drive from A to B on



- Set the social cost to be the maximum travel time.

Braess' Network

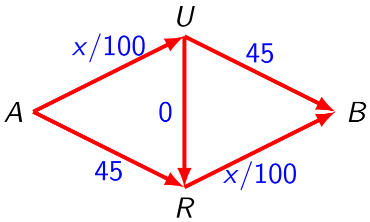
- 4000 drivers drive from A to B on



- Set the social cost to be the maximum travel time.
- Optimal social cost is reached when half of the drivers take $A - U - B$ and the other half $A - R - B$ with social cost 65.

Braess' Network

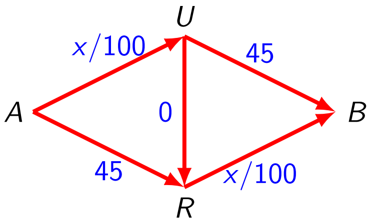
- 4000 drivers drive from A to B on



- Set the social cost to be the maximum travel time.
- Optimal social cost is reached when half of the drivers take $A - U - B$ and the other half $A - R - B$ with social cost 65.
- In the NE

Braess' Network

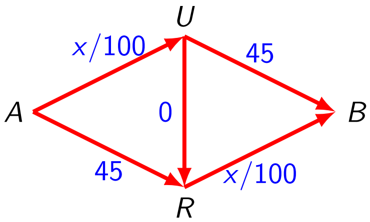
- 4000 drivers drive from A to B on



- Set the social cost to be the maximum travel time.
- Optimal social cost is reached when half of the drivers take $A-U-B$ and the other half $A-R-B$ with social cost 65.
- In the NE all drivers take $A-U-R-B$ with social cost 80.

Braess' Network

- 4000 drivers drive from A to B on



- Set the social cost to be the maximum travel time.
- Optimal social cost is reached when half of the drivers take $A-U-B$ and the other half $A-R-B$ with social cost 65.
- In the NE all drivers take $A-U-R-B$ with social cost 80.
- $PoA = PoS = 80/65 = 16/13$

Load Balancing game

- There are m servers and n jobs. Job i has load p_i.
- The game has n players, corresponding to the n jobs.
- Each player has to decide the server that will process its job. $A_i = \{1, \ldots, m\}$
- The response time of server j is proportional to its load

$$L_j(s) = \sum_{i|s_i=j} p_i.$$

- Each job wants to be assigned to the server that minimizes its response time:

$$c_i(s) = L_{s_i}(s).$$

Load Balancing game: PNE?

Consider the best response dynamic

- Start with an arbitrary state.
- A node (or several) chooses a best strategy, one that maximizes its own payoff, given the current choices of the others

Load Balancing game: PNE?

Consider the best response dynamic

- Start with an arbitrary state.
- A node (or several) chooses a best strategy, one that maximizes its own payoff, given the current choices of the others

- How to prove that such a process converges to a PNE?

Load Balancing game: PNE?

Consider the best response dynamic

- Start with an arbitrary state.
- A node (or several) chooses a best strategy, one that maximizes its own payoff, given the current choices of the others

- How to prove that such a process converges to a PNE?
- Seek for an adequate kind of potential function.

Load Balancing game: PNE?

BR-inspired-algorithm

- Order the servers with decreasing load (i.e., the decreasing response time):
$$L_1 \geq L_2 \geq \cdots \geq L_m.$$
- Job i moves from server j to k, $L_k + p_i < L_j$.
- We must have $L_1 \geq \cdots \geq L_j \geq \cdots \geq L_k \geq \cdots \geq L_m$.
- Thus, $L_j - p_i < L_j$ and $L_k + p_i < L_j$.

Load Balancing game: PNE?

BR-inspired-algorithm

- Order the servers with decreasing load (i.e., the decreasing response time):
$$L_1 \geq L_2 \geq \cdots \geq L_m.$$
- Job i moves from server j to k, $L_k + p_i < L_j$.
- We must have $L_1 \geq \cdots \geq L_j \geq \cdots \geq L_k \geq \cdots \geq L_m$.
- Thus, $L_j - p_i < L_j$ and $L_k + p_i < L_j$.
- Reorder the servers by decreasing load and repeat the process until no job can move.

Load Balancing game: PNE?

- Does the algorithm converge?

Load Balancing game: PNE?

- Does the algorithm converge?
- There are a finite number of (possibly exponential) assignments of jobs to servers.

Load Balancing game: PNE?

- Does the algorithm converge?
- There are a finite number of (possibly exponential) assignments of jobs to servers.
- At each step the sorted load sequence decreases lexicographically!

Load Balancing game: PNE?

- Does the algorithm converge?
- There are a finite number of (possibly exponential) assignments of jobs to servers.
- At each step the sorted load sequence decreases lexicographically!

  The number of machines with load $< L_j$ decreases

Load Balancing game: PNE?

- Does the algorithm converge?
- There are a finite number of (possibly exponential) assignments of jobs to servers.
- At each step the sorted load sequence decreases lexicographically!

  The number of machines with load $< L_j$ decreases
- So BR-inspired-algorithm terminates (although it can be rather slow).

Load Balancing game: PNE?

- Does the algorithm converge?
- There are a finite number of (possibly exponential) assignments of jobs to servers.
- At each step the sorted load sequence decreases lexicographically!

  The number of machines with load $< L_j$ decreases
- So BR-inspired-algorithm terminates (although it can be rather slow).
- The load balancing game has a PNE.

Load Balancing game: Social cost

- The natural social cost is the total finish time i.e., the maximum of the server's loads

$$c(s) = \max_{j=1}^{m} L_j.$$

- How bad/good is a PNE?

Load Balancing game: PoS

- Let s be an assignment with optimal cost.
- Is s a PNE?

Load Balancing game: PoS

- Let s be an assignment with optimal cost.
- Is s a PNE?
- Not necessarily, no player in the worst server can improve, but other players can get a better benefit.

Load Balancing game: PoS

- Let s be an assignment with optimal cost.
- Is s a PNE?
- Not necessarily, no player in the worst server can improve, but other players can get a better benefit.
- However, starting from an optimal solution the BR-inspired-algorithm terminates on a PNE with the same maximum load.

Load Balancing game: PoS

- Let s be an assignment with optimal cost.
- Is s a PNE?
- Not necessarily, no player in the worst server can improve, but other players can get a better benefit.
- However, starting from an optimal solution the BR-inspired-algorithm terminates on a PNE with the same maximum load.
- Therefore, $PoS(\Gamma) = 1$.

Load Balancing game: PoA

Theorem

The max load of a Nash equilibrium s is within twice the max load of an optimum assignment, i.e.,.

$$C(s) \leq 2 \min_{s'} C(s').$$

Which will give $PoA(\Gamma) \leq 2$.

Load Balancing game: PoA bound

Load Balancing game: PoA bound

- Let s be a PNE
- Let i be a job assigned to the max loaded server j.

Load Balancing game: PoA bound

- Let s be a PNE
- Let i be a job assigned to the max loaded server j.
  - $L_j \leq L_k + p_i$, for all other server k.

Load Balancing game: PoA bound

- Let s be a PNE
- Let i be a job assigned to the max loaded server j.
  - $L_j \leq L_k + p_i$, for all other server k.
  - Summing over all servers, we get

Load Balancing game: PoA bound

- Let s be a PNE
- Let i be a job assigned to the max loaded server j.
  - $L_j \leq L_k + p_i$, for all other server k.
  - Summing over all servers, we get $L_j \leq (\sum_k L_k)/m + p_i$.

Load Balancing game: PoA bound

- Let s be a PNE
- Let i be a job assigned to the max loaded server j.
  - $L_j \leq L_k + p_i$, for all other server k.
  - Summing over all servers, we get $L_j \leq (\sum_k L_k)/m + p_i$.
- In an opt solution, i is assigned to some server,

Load Balancing game: PoA bound

- Let s be a PNE
- Let i be a job assigned to the max loaded server j.
  - $L_j \leq L_k + p_i$, for all other server k.
  - Summing over all servers, we get $L_j \leq (\sum_k L_k)/m + p_i$.
- In an opt solution, i is assigned to some server, so $C(s') \geq p_i$.

Load Balancing game: PoA bound

- Let s be a PNE
- Let i be a job assigned to the max loaded server j.
  - $L_j \leq L_k + p_i$, for all other server k.
  - Summing over all servers, we get $L_j \leq (\sum_k L_k)/m + p_i$.
- In an opt solution, i is assigned to some server, so $C(s') \geq p_i$.
- $\sum_k L_k$ is the total processing time for an assignment.

Load Balancing game: PoA bound

- Let s be a PNE
- Let i be a job assigned to the max loaded server j.
  - $L_j \leq L_k + p_i$, for all other server k.
  - Summing over all servers, we get $L_j \leq (\sum_k L_k)/m + p_i$.
- In an opt solution, i is assigned to some server, so $C(s') \geq p_i$.
- $\sum_k L_k$ is the total processing time for an assignment.
  The best possible algorithm is to evenly partition them among m servers (if possible), thus

Load Balancing game: PoA bound

- Let s be a PNE
- Let i be a job assigned to the max loaded server j.
  - $L_j \leq L_k + p_i$, for all other server k.
  - Summing over all servers, we get $L_j \leq (\sum_k L_k)/m + p_i$.
- In an opt solution, i is assigned to some server, so $C(s') \geq p_i$.
- $\sum_k L_k$ is the total processing time for an assignment.
  The best possible algorithm is to evenly partition them among m servers (if possible), thus $C(s') \geq \sum_k L_k/m = (\sum_\ell p_\ell)/m$.

Load Balancing game: PoA bound

- Let s be a PNE
- Let i be a job assigned to the max loaded server j.
  - $L_j \leq L_k + p_i$, for all other server k.
  - Summing over all servers, we get $L_j \leq (\sum_k L_k)/m + p_i$.
- In an opt solution, i is assigned to some server, so $C(s') \geq p_i$.
- $\sum_k L_k$ is the total processing time for an assignment.
  The best possible algorithm is to evenly partition them among m servers (if possible), thus $C(s') \geq \sum_k L_k/m = (\sum_\ell p_\ell)/m$.
- We get
  $C(s) = L_j \leq (\sum_k L_k)/m + p_i$

Load Balancing game: PoA bound

- Let s be a PNE
- Let i be a job assigned to the max loaded server j.
  - $L_j \leq L_k + p_i$, for all other server k.
  - Summing over all servers, we get $L_j \leq (\sum_k L_k)/m + p_i$.
- In an opt solution, i is assigned to some server, so $C(s') \geq p_i$.
- $\sum_k L_k$ is the total processing time for an assignment.
  The best possible algorithm is to evenly partition them among m servers (if possible), thus $C(s') \geq \sum_k L_k/m = (\sum_\ell p_\ell)/m$.
- We get
  $C(s) = L_j \leq (\sum_k L_k)/m + p_i \leq (\sum_\ell p_\ell)/m + p_i$

Load Balancing game: PoA bound

- Let s be a PNE
- Let i be a job assigned to the max loaded server j.
    - $L_j \leq L_k + p_i$, for all other server k.
    - Summing over all servers, we get $L_j \leq (\sum_k L_k)/m + p_i$.
- In an opt solution, i is assigned to some server, so $C(s') \geq p_i$.
- $\sum_k L_k$ is the total processing time for an assignment.
  The best possible algorithm is to evenly partition them among m servers (if possible), thus $C(s') \geq \sum_k L_k/m = (\sum_\ell p_\ell)/m$.
- We get
  $$C(s) = L_j \leq (\sum_k L_k)/m + p_i \leq (\sum_\ell p_\ell)/m + p_i$$
  $$\leq C(s') + C(s').$$

Congestion games

Congestion games

A congestion game $(E, N, (d_e)_{e \in E})$

- is defined on a finite set E of resources and
- has n players and,
- for each resource e, a delay function d_e mapping $\mathbb{N}$ to the integers.
- The actions for each player are subsets of E.
- The cost functions are the following:

$$c_i(a_1, \dots, a_n) = \left(\sum_{e \in a_i} d(e, f(a_1, \dots, a_n, e)) \right)$$

being $f_e(a_1, \dots, a_n, e) = |\{i \mid e \in a_i\}|$.

Weighted congestion games

Weighted congestion games

A weighted congestion game $(E, N, (d_e)_{e \in E}, (w_i)_{i \in N})$

- is defined on a finite set E of resources and
- has n players. Player i has an associated positive integer weight w_i.
- Each resource e has a delay function d_e mapping $\mathbb{N}$ to the integers.
- The actions for each player are subsets of E.
- The cost functions are the following:

$$c_i(a_1, \ldots, a_n) = \left(\sum_{e \in a_i} d(e, f(a_1, \ldots, a_n, e)) \right)$$

being $f_e(a_1, \ldots, a_n, e) = \sum_{i | e \in a_i} w_i$.

Network weighted congestion games

Network weighted congestion games

A network weighted congestion game $(N, G = (V, E), (d_e)_{e \in E}, (w_i)_{i \in N}, (s_i)_{i \in N}, (t_i)_{i \in N})$

- Is defined on a directed graph $G = (V, E)$, the resources are the arcs (E)
- The game has n players, player i has an associated positive integer weight w_i and two vertices $s_i, t_i \in V$.
- For each arc e a delay function d_e mapping $\mathbb{N}$ to the integers.
- The action set for player i is the set of $(s_i - t_i)$-paths in G.
- The cost functions are the following:

$$c_i(a_1, \ldots, a_n) = \left(\sum_{e \in a_i} d(e, f(a_1, \ldots, a_n, e)) \right)$$

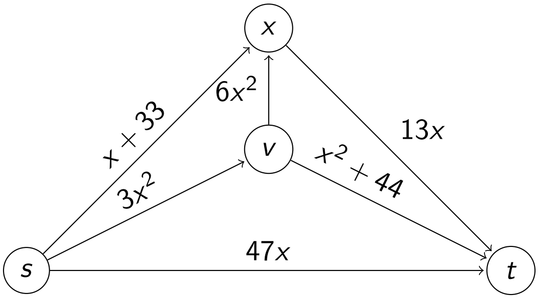
being $f(a_1, \ldots, a_n, e) = \sum_{i | e \in a_i} w_i$.

PNE in weighted congestion games

- There are weighted network congestion games without PNE

PNE in weighted congestion games

- There are weighted network congestion games without PNE
- Consider the following network with 2 players having weights $w_1 = 1$ and $w_2 = 2$.



Not always PNE in weighted congestion games

Not always PNE in weighted congestion games

s_{-i}	BR_1	BR_2
$P_1 : s \to t$	P_4	P_2
$P_2 : s \to v \to t$	P_4	P_4
$P_3 : s \to w \to t$	P_1	P_2
$P_3 : s \to v \to w \to t$	P_1	P_3

Not always PNE in weighted congestion games

s_{-i}	BR_1	BR_2
$P_1 : s \rightarrow t$	P_4	P_2
$P_2 : s \rightarrow v \rightarrow t$	P_4	P_4
$P_3 : s \rightarrow w \rightarrow t$	P_1	P_2
$P_3 : s \rightarrow v \rightarrow w \rightarrow t$	P_1	P_3

Therefore the game has no PNE

PoA for affine congestion games

Consider unweighted congestion games such that the delay functions are affine functions, i.e., for each resource e,

$$d_e(x) = a_e x + b_e,$$

for some $a_e, b_e > 0$.

PoA for affine congestion games

Consider unweighted congestion games such that the delay functions are affine functions, i.e., for each resource e,

$$d_e(x) = a_e x + b_e,$$

for some $a_e, b_e > 0$.

Let C be the usual social cost:

$$C(s) = \sum_{e \in E} d_e(f_e(s))$$

Smoothness

A game is called (λ, μ)-smooth, for $\lambda > 0$ and $\mu \leq 1$ if, for every pair of strategy profiles s and s', we have

$$\sum_{i \in N} c_i(s_{-i}, s'_i) \leq \lambda C(s') + \mu C(s).$$

Smoothness

A game is called (λ, μ)-smooth, for $\lambda > 0$ and $\mu \leq 1$ if, for every pair of strategy profiles s and s', we have

$$\sum_{i \in N} c_i(s_{-i}, s'_i) \leq \lambda C(s') + \mu C(s).$$

Smoothness directly gives a bound for the PoA:

Smoothness

A game is called (λ, μ)-smooth, for $\lambda > 0$ and $\mu \leq 1$ if, for every pair of strategy profiles s and s', we have

$$\sum_{i \in N} c_i(s_{-i}, s'_i) \leq \lambda C(s') + \mu C(s).$$

Smoothness directly gives a bound for the PoA:

Theorem

In a (λ, μ)-smooth game, the PoA for PNE is at most $\frac{\lambda}{1-\mu}$.

Proof of smoothness bound on PoA

Let s be the worst PNE and s^* be an optimum solution.

$$C(s) = \sum_{i \in N} c_i(s) \leq \sum_{i \in N} c_i(s_{-i}, s_i^*)$$
$$\leq \lambda C(s^*) + \mu C(s)$$

Substracting $\mu C(s)$ on both sides gives

$$(1-\mu)C(s) \leq \lambda C(s^*).$$

Theorem

Every congestion game with affine delay functions is $(5/3, 1/3)$-smooth. Thus, PoA $\leq 5/2$.

Theorem

Every congestion game with affine delay functions is $(5/3, 1/3)$-smooth. Thus, $PoA \leq 5/2$.

The proof uses a technical lemma:

Lemma (Christodoulou, Koutsoupias, 2005)

For all integers y, z we have

$$y(z+1) \leq \frac{5}{3}y^2 + \frac{1}{3}z^2.$$

Proof of smoothness for affine functions

Recall that $d_e(x) = a_e x + b_e$. Note that using the Lemma

$$a_e y(z+1)+b_e y \leq a_e(\frac{5}{3}y^2+\frac{1}{3}z^2)+b_e y = \frac{5}{3}(a_e y^2+b_e y)+\frac{1}{3}(a_e z^2+b_e z).$$

Proof of smoothness for affine functions

Recall that $d_e(x) = a_e x + b_e$. Note that using the Lemma

$$a_e y(z+1)+b_e y \leq a_e(\frac{5}{3}y^2+\frac{1}{3}z^2)+b_e y = \frac{5}{3}(a_e y^2+b_e y)+\frac{1}{3}(a_e z^2+b_e z).$$

Taking $y = f_e(s^*)$ and $z = f_e(s)$ we get

$$(a_e(f_e(s)+1)+b_e)f_e(s^*) \leq \frac{5}{3}(a_e f_e(s^*)+b_e)f_e(s^*))+\frac{1}{3}(a_e f_e(s)+b_e)f_e(s)).$$

Proof of smoothness for affine functions

Recall that $d_e(x) = a_e x + b_e$. Note that using the Lemma

$$a_e y(z+1)+b_e y \leq a_e(\frac{5}{3}y^2+\frac{1}{3}z^2)+b_e y = \frac{5}{3}(a_e y^2+b_e y)+\frac{1}{3}(a_e z^2+b_e z).$$

Taking $y = f_e(s^*)$ and $z = f_e(s)$ we get

$$(a_e(f_e(s)+1)+b_e)f_e(s^*) \leq \frac{5}{3}(a_e f_e(s^*)+b_e)f_e(s^*))+\frac{1}{3}(a_e f_e(s)+b_e)f_e(s)).$$

Summing up all the inequalities

$$\sum_{e \in E}(a_e(f_e(s)+1)+b_e)f_e(s^*) \leq \frac{5}{3}C(s^*)+\frac{1}{3}C(s).$$

Proof of smoothness for affine functions

$$\sum_{e \in E} (a_e(f_e(s)+1)+b_e) f_e(s^*) \leq \frac{5}{3} C(s^*) + \frac{1}{3} C(s).$$

Proof of smoothness for affine functions

$$\sum_{e \in E}(a_e(f_e(s)+1)+b_e)f_e(s^*) \leq \frac{5}{3}C(s^*)+\frac{1}{3}C(s).$$

But,

$$\sum_{i \in N} c_i(s_{-i}, s_i^*) \leq \sum_{e \in E}(a_e(f_e(s)+1)+b_e)f_e(s^*)$$

as there are at most $f_e(s^*)$ players that might move to resource r. Each of them by unilaterally deviating incur a delay of $(a_e(f_e(s)+1)+b_e$.

Proof of smoothness for affine functions

$$\sum_{e \in E} (a_e(f_e(s)+1)+b_e)f_e(s^*) \leq \frac{5}{3}C(s^*) + \frac{1}{3}C(s).$$

But,

$$\sum_{i \in N} c_i(s_{-i}, s_i^*) \leq \sum_{e \in E} (a_e(f_e(s)+1)+b_e)f_e(s^*)$$

as there are at most $f_e(s^*)$ players that might move to resource r. Each of them by unilaterally deviating incur a delay of $(a_e(f_e(s)+1)+b_e$.
This gives the $(5/3, 1/3)$-smoothness.

References

- Chapters 18 and 19.3 in the AGT book. (PoA and PoS bounds).
- B. Awerbuch, Y. Azar, A. Epstein. The Price of Routing Unsplittable Flow. STOC 2005. (PoA for pure NE in congestion games).
- G. Christodoulou, E. Koutsoupias. The Price of Anarchy of finite Congestion Games. STOC 2005. (PoA for pure NE in congestion games)
- T. Roughgarden. Intrinsic Robustness of the Price of Anarchy. STOC 2009. (Smoothness Framework and Unification of Previous Results)
- D. Fotakis. A Selective Tour Through Congestion Games, LNCS 2015.