Efficiency of Nash Equilibria

Maria Serna

Fall 2019

Efficiency at equilibrium

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

Efficiency at equilibrium

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

Efficiency at equilibrium

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

Efficiency at equilibrium

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

Social cost

- Consider a n-player game $\Gamma = (A_1, \ldots, A_n, u_1, \ldots, u_n)$.
- Let $A = A_1 \times \cdots \times A_n$.
- Let $PNE(\Gamma)$ be the set of PNE of Γ.
- Let $NE(\Gamma)$ be the set of NE of Γ.

Social cost

- Consider a n-player game $\Gamma = (A_1, \ldots, A_n, u_1, \ldots, u_n)$.
- Let $A = A_1 \times \cdots \times A_n$.
- Let $PNE(\Gamma)$ be the set of PNE of Γ.
- Let $NE(\Gamma)$ be the set of NE of Γ.
- Let $\mathcal{C} : A \to \mathbb{R}$ be a social cost function.

  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 : $C(s) = \sum_{i \in N} u_i(s)$.

Usual social cost functions

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

Usual social cost functions

- Utilitarian social cost : $C(s) = \sum_{i \in N} u_i(s)$.
- Egalitarian social cost: $C(s) = \max_{i \in N} u_i(s)$.
- 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, the one giving the maximum system degradation.

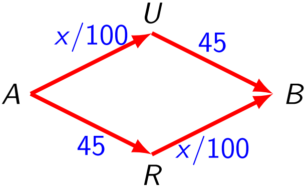
Price of Anarchy/Stability

- 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, the one giving the maximum system degradation.
- PoS measures the best decentralized equilibrium scenario, the one giving the best possible degradation.

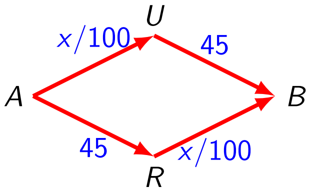
Network congestion game

- 4000 drivers drive from A to B on



Network congestion game

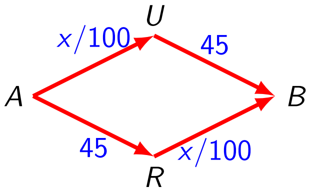
- 4000 drivers drive from A to B on



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

Network congestion game

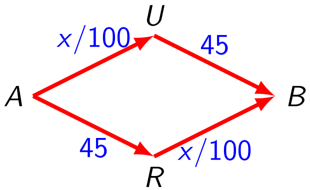
- 4000 drivers drive from A to B on



- 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 congestion game

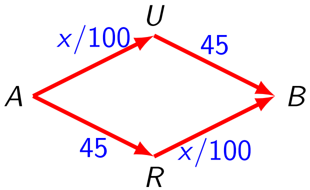
- 4000 drivers drive from A to B on



- 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 congestion game

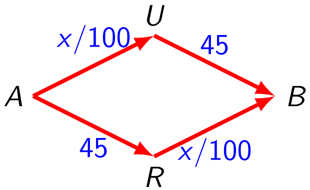
- 4000 drivers drive from A to B on



- 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 half of the drivers take $A - U - B$ and the other half $A - R - B$.

Network congestion game

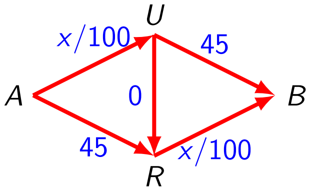
- 4000 drivers drive from A to B on



- 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 half of the drivers take $A - U - B$ and the other half $A - R - B$.
- $PoA = PoS = 65/65 = 1$

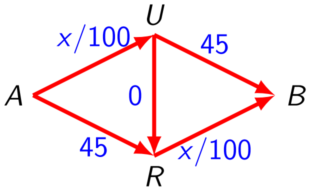
Braess' Network

- 4000 drivers drive from A to B on



Braess' Network

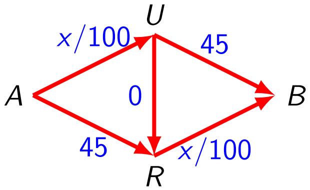
- 4000 drivers drive from A to B on



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

Braess' Network

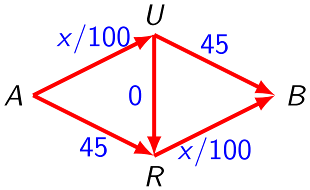
- 4000 drivers drive from A to B on



- 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

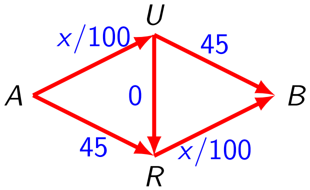
- 4000 drivers drive from A to B on



- 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

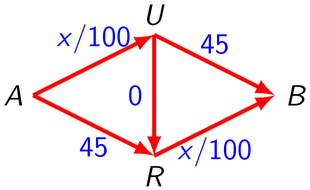
- 4000 drivers drive from A to B on



- 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



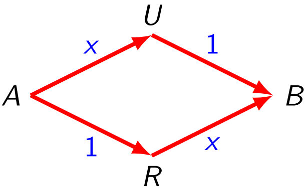
- Set the social cost to be the maximum travel time.
- Optimal social cost is reached when half of the drivers take $A - U - B$ and the other half $A - R - B$ with social cost 65.
- In the NE all drivers take $A - U - R - B$ with social cost 80.
- $PoA = PoS = 80/65 = 16/13$

Selfish Routing: an Example

- Total traffic is $r = 1$.
- Network (with delay functions on arcs)

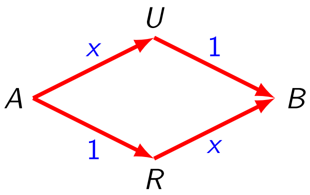
Selfish Routing: an Example

- Total traffic is $r = 1$.
- Network (with delay functions on arcs)



Selfish Routing: an Example

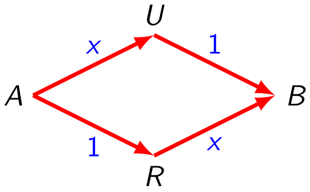
- Total traffic is $r = 1$.
- Network (with delay functions on arcs)



- Player's objective going from $s = A$ to $s = B$ with minimum delay.

Selfish Routing: an Example

- Total traffic is $r = 1$.
- Network (with delay functions on arcs)



- Player's objective going from $s = A$ to $s = B$ with minimum delay.
- Strategy profiles: flows from A to B with total flow $r = 1$

Selfish routing: strategy profiles

Selfish routing: strategy profiles

- Traffic as Flows:

Selfish routing: strategy profiles

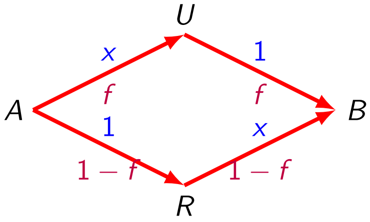
- Traffic as Flows:
  - A flow f giving the routing of traffic.

Selfish routing: strategy profiles

- Traffic as Flows:
  - A flow f giving the routing of traffic.
  - Recall that a flow must preserve flow in $=$ flow out except for sources/sinks.

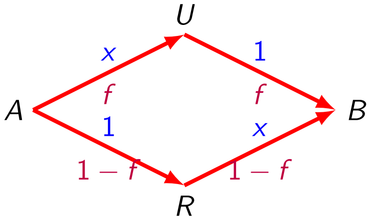
Selfish routing: strategy profiles

- Traffic as Flows:
  - A flow f giving the routing of traffic.
  - Recall that a flow must preserve flow in = flow out except for sources/sinks.



Selfish routing: strategy profiles

- Traffic as Flows:
  - A flow f giving the routing of traffic.
  - Recall that a flow must preserve flow in = flow out except for sources/sinks.



- Notation: for a path P and a feasible flow f, $C^P(f)$ denotes the cost corresponding to the traffic routed through P by f.

Selfish routing: equilibria

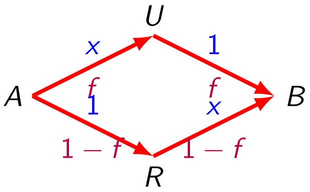
Theorem

A flow is a Nash equilibrium (or is a Nash flow) if all flow is routed on min-latency paths (given current edge congestion)

Selfish routing: equilibria

Theorem

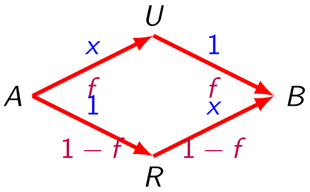
A flow is a Nash equilibrium (or is a Nash flow) if all flow is routed on min-latency paths (given current edge congestion)



Selfish routing: equilibria

Theorem

A flow is a Nash equilibrium (or is a Nash flow) if all flow is routed on min-latency paths (given current edge congestion)



For f to be a Nash equilibrium, all $A - B$ paths should have minimum latency, so $f = 1/2$.

Selfish routing: equilibria

Selfish routing: equilibria

Theorem

A feasible flow x is an equilibrium flow iff for any feasible flow y

$$\sum_{e \in E} d_e(x[e])x[e] \leqslant \sum_{e \in E} d_e(x[e])y[e].$$

Selfish routing: equilibria

Theorem

A feasible flow x is an equilibrium flow iff for any feasible flow y

$$\sum_{e \in E} d_e(x[e])x[e] \leqslant \sum_{e \in E} d_e(x[e])y[e].$$

- Called Variational Inequality (Smith 79 and Dafermos 80)

Selfish routing: equilibria

Theorem

A feasible flow x is an equilibrium flow iff for any feasible flow y

$$\sum_{e \in E} d_e(x[e])x[e] \leqslant \sum_{e \in E} d_e(x[e])y[e].$$

- Called Variational Inequality (Smith 79 and Dafermos 80)
- As a consequence all Nash flows have the same cost per edge.

Selfish routing: equilibria

Theorem

A feasible flow x is an equilibrium flow iff for any feasible flow y

$$\sum_{e \in E} d_e(x[e])x[e] \leqslant \sum_{e \in E} d_e(x[e])y[e].$$

- Called Variational Inequality (Smith 79 and Dafermos 80)
- As a consequence all Nash flows have the same cost per edge.
- Do PNE exist?

Selfish routing: equilibria existence

- As for the atomic case we can consider a potential, for a given flow x

$$\Psi(x) = \sum_{e \in E} \int_0^{x[e]} d_e(u) du$$

Selfish routing: equilibria existence

- As for the atomic case we can consider a potential, for a given flow x

$$\Psi(x) = \sum_{e \in E} \int_0^{x[e]} d_e(u) du$$

Theorem

A feasible flow x is an equilibrium flow iff x is a minimum of Ψ over the set of feasible flows.

Selfish routing

- Social cost: maximum travel time egalitarian
  - By the characterization of Nash flows all NE have the same cost.
  - PoA=PoS = cost NE / opt

Selfish routing

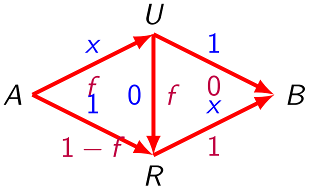
- Social cost: maximum travel time — egalitarian
  - By the characterization of Nash flows all NE have the same cost.
  - PoA=PoS = cost NE / opt
- Other social cost?
  A natural one is the total travel time — utilitarian

Selfish routing: total routing time

The cost $C(f)$ of flow f is the sum of all delays incurred by traffic.

Selfish routing: total routing time

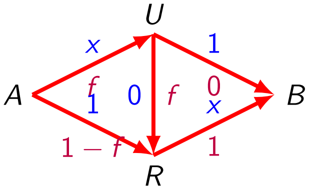
The cost $C(f)$ of flow f is the sum of all delays incurred by traffic.



$C(f) = f(f+1) + (1-f)2.$

Selfish routing: total routing time

The cost $C(f)$ of flow f is the sum of all delays incurred by traffic.



$C(f) = f(f+1) + (1-f)2.$

Formally, if $d_P(f)$ is the sum of latencies of edges in a path P :

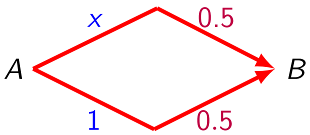
$$C(f) = \sum_P f_P\, d_P(f)$$

Selfish routing: inefficiency of Nash flows

- Nash flows do not minimize total latency

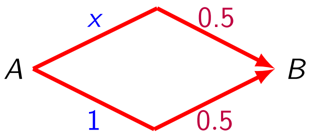
Selfish routing: inefficiency of Nash flows

- Nash flows do not minimize total latency



Selfish routing: inefficiency of Nash flows

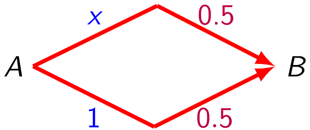
- Nash flows do not minimize total latency



- Cost: $f^2 + 1 - f + 0.5$

Selfish routing: inefficiency of Nash flows

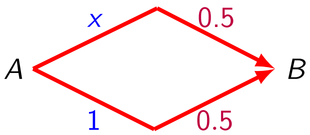
- Nash flows do not minimize total latency



- Cost: $f^2 + 1 - f + 0.5$
- Optimal cost $0.25 + 0.5 + 0.5 = 1.25$

Selfish routing: inefficiency of Nash flows

- Nash flows do not minimize total latency



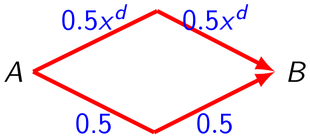
- Cost: $f^2 + 1 - f + 0.5$
- Optimal cost $0.25 + 0.5 + 0.5 = 1.25$
- Nash flow has cost 1.5

Selfish routing: inefficiency of Nash flows

- An extreme case:

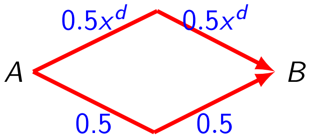
Selfish routing: inefficiency of Nash flows

- An extreme case:



Selfish routing: inefficiency of Nash flows

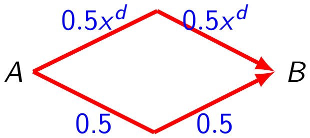
- An extreme case:



- Cost: $(1 - f)^d + f$

Selfish routing: inefficiency of Nash flows

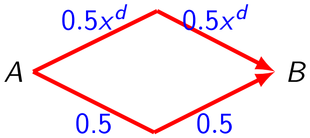
- An extreme case:



- Cost: $(1-f)^d + f$
- For d large enough, as $r = 1$, optimal cost $\epsilon + (1-\epsilon)^d$ where $\epsilon \to 0$ as $d \to \infty$

Selfish routing: inefficiency of Nash flows

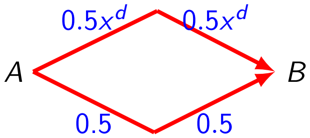
- An extreme case:



- Cost: $(1-f)^d + f$
- For d large enough, as $r = 1$, optimal cost $\epsilon + (1-\epsilon)^d$ where $\epsilon \to 0$ as $d \to \infty$
- Nash flow has cost 1

Selfish routing: inefficiency of Nash flows

- An extreme case:



- Cost: $(1-f)^d + f$
- For d large enough, as $r = 1$, optimal cost $\epsilon + (1-\epsilon)^d$ where $\epsilon \to 0$ as $d \to \infty$
- Nash flow has cost 1
- Unbounded PoA: Nash flow can cost arbitrarily more than the optimal (mincost) flow even if latency functions are polynomials

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. $A_i = \{1, \dots, m\}$
- The response time of server j is proportional to its load

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

- 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):
$$L_1 \geqslant L_2 \geqslant \ldots \geqslant L_m.$$
- Job i moves from server j to k, $L_k + p_i < L_j$.
- We must have $L_1 \geqslant \ldots \geqslant L_j \geqslant \ldots \geqslant L_k \geqslant \ldots \geqslant L_m$.
- Thus, $L_j - p_i, L_k + p_i < L_j$

Load Balancing game: PNE?

BR-inspired-algorithm

- Order the servers with decreasing load (i.e., the decreasing response time):
$$L_1 \geqslant L_2 \geqslant \ldots \geqslant L_m.$$
- Job i moves from server j to k, $L_k + p_i < L_j$.
- We must have $L_1 \geqslant \ldots \geqslant L_j \geqslant \ldots \geqslant L_k \geqslant \ldots \geqslant L_m$.
- Thus, $L_j - p_i, L_k + p_i < L_j$
- 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!
- 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!
- 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, however 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, however 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, however 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 Pure Nash equilibrium s is within twice the max load of an optimum assignment, i.e.,.

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

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

Load Balancing game: PoA bound

Load Balancing game: PoA bound

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

Load Balancing game: PoA bound

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

Load Balancing game: PoA bound

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

Load Balancing game: PoA bound

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

Load Balancing game: PoA bound

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

Load Balancing game: PoA bound

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

Load Balancing game: PoA bound

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

Load Balancing game: PoA bound

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

Load Balancing game: PoA bound

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

Load Balancing game: PoA bound

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

Congestion games

Congestion games

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

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

$$c_i(a_1, \dots, a_n) = \sum_{e\in a_i} d_e(f_e(a_1, \dots, a_n))$$

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

Weighted congestion games

Weighted congestion games

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

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

$$c_i(a_1,\ldots,a_n) = \sum_{e\in a_i} d_e(f_e(a_1,\ldots,a_n))$$

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

Network weighted congestion games

Network weighted congestion games

A network weighted congestion game is defined on a directed graph $G = (V.E)$, $(N, G, (d_e)_{e\in E}, (c_i)_{i\in N}, (w_i)_{i\in N}, (s_i)_{i\in N}, (t_i)_{i\in N})$.

- The resources are the arcs in G.
- The game has n players. Player i has an associated natural weight w_i.
- Using a delay function d_e mapping $\mathbb{N}$ to the integers, for each arc $e \in E$.
- The action set for player i is the set of (s_i, t_i)-paths in G.
- The cost functions are the following:

$$c_i(a_1, \dots, a_n) = \sum_{e\in a_i} d_e(f_e(a_1, \dots, a_n))$$

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

Another family: Fair Cost Sharing Games

Another family: Fair Cost Sharing Games

A fair cost sharing game $(E, N, (c_e)_{e \in E})$

- is defined on a finite set E of resources and
- has n players
- a fixed cost c_e, for each resource e.
- The actions for each player are subsets of E.
- The cost functions are the following:

$$c_i(a_1, \dots, a_n) = \sum_{e \in a_i} \frac{c_e}{f_e(a_1, \dots, a_n)}$$

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

Congestion games terminology

Congestion games terminology

- unweighted (vs. weighted)

Congestion games terminology

- unweighted (vs. weighted): $w_i = 1$.

Congestion games terminology

- unweighted (vs. weighted): $w_i = 1$.
- symmetric (vs. non-symmetric) strategies:

Congestion games terminology

- unweighted (vs. weighted): $w_i = 1$.
- symmetric (vs. non-symmetric) strategies: all the players have the same set of actions.

Congestion games terminology

- unweighted (vs. weighted): $w_i = 1$.
- symmetric (vs. non-symmetric) strategies: all the players have the same set of actions.
- symmetric congestion games:

Congestion games terminology

- unweighted (vs. weighted): $w_i = 1$.
- symmetric (vs. non-symmetric) strategies: all the players have the same set of actions.
- symmetric congestion games: unweighted with symmetric strategies.

Congestion games terminology

- unweighted (vs. weighted): $w_i = 1$.
- symmetric (vs. non-symmetric) strategies: all the players have the same set of actions.
- symmetric congestion games: unweighted with symmetric strategies.
- singleton congestion games:

Congestion games terminology

- unweighted (vs. weighted): $w_i = 1$.
- symmetric (vs. non-symmetric) strategies: all the players have the same set of actions.
- symmetric congestion games: unweighted with symmetric strategies.
- singleton congestion games: all possible actions have only one resource.

Congestion games terminology

- unweighted (vs. weighted): $w_i = 1$.
- symmetric (vs. non-symmetric) strategies: all the players have the same set of actions.
- symmetric congestion games: unweighted with symmetric strategies.
- singleton congestion games: all possible actions have only one resource.
- nonatomic network congestion games (vs. atomic)

Congestion games terminology

- unweighted (vs. weighted): $w_i = 1$.
- symmetric (vs. non-symmetric) strategies: all the players have the same set of actions.
- symmetric congestion games: unweighted with symmetric strategies.
- singleton congestion games: all possible actions have only one resource.
- nonatomic network congestion games (vs. atomic)
  In nonatomic congestion games the number of players is infinite and each player controls an infinitesimal weight of the total traffic.

Congestion games terminology

- unweighted (vs. weighted): $w_i = 1$.
- symmetric (vs. non-symmetric) strategies: all the players have the same set of actions.
- symmetric congestion games: unweighted with symmetric strategies.
- singleton congestion games: all possible actions have only one resource.
- nonatomic network congestion games (vs. atomic)
  In nonatomic congestion games the number of players is infinite and each player controls an infinitesimal weight of the total traffic. Named also Selfish routing games.

PNE in Weighted Congestion Games

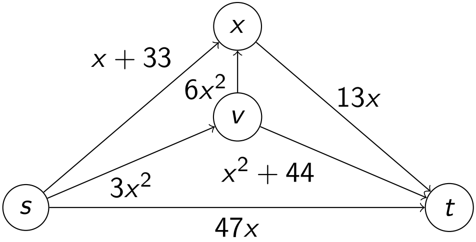
- There are weighted network congestion games without PNE

PNE in Weighted Congestion Games

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

PNE in Weighted Congestion Games

- There are weighted network congestion games without PNE
- 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
$P_4 : s \to v \to w \to t$	P_1	P_3

Not always PNE in Weighted Congestion Games

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

Therefore the game has no PNE

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 $a_e, b_e > 0$.

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:

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

PNE in Affine Congestion Games

PNE in Affine Congestion Games

- For affine delay functions PNE always exist

PNE in Affine Congestion Games

- For affine delay functions PNE always exist
  Show that the following $\Phi(s)$ is a weighted potential function

PNE in Affine Congestion Games

- For affine delay functions PNE always exist
  Show that the following $\Phi(s)$ is a weighted potential function

$$U(s) = \sum_{i \in N} w_i \sum_{e \in s_i} (a_e w_i + b_e) \qquad C(s) = \sum_{i \in N} w_i c_i(s)$$

$$\Phi(s) = (C(s) + U(s))/2.$$

PNE in Affine Congestion Games

- For affine delay functions PNE always exist
  Show that the following $\Phi(s)$ is a weighted potential function

$$U(s) = \sum_{i \in N} w_i \sum_{e \in s_i} (a_e w_i + b_e) \qquad C(s) = \sum_{i \in N} w_i c_i(s)$$

$$\Phi(s) = (C(s) + U(s))/2.$$

You should be able to show that

$$\Phi(s') - \Phi(s) = w_i(c_i(s') - c_i(s)).$$

Smoothness

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

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

Smoothness

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

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

Smoothness directly gives a bound for the PoA:

Smoothness

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

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

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.

$$\begin{aligned} C(s) = \sum_{i \in N} c_i(s) &\leqslant \sum_{i \in N} c_i(s_{-i}, s_i^*) \\ &\leqslant \lambda C(s^*) + \mu C(s) \end{aligned}$$

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

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

Theorem

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

Theorem

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

The proof uses a technical lemma:

Lemma (Christodoulou, Koutsoupias, 2005)

For all integers y, z we have

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

Proof of smoothness for affine functions

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

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

Proof of smoothness for affine functions

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

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

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

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

Proof of smoothness for affine functions

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

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

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

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

Summing up all the inequalities

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

Proof of smoothness for affine functions

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

Proof of smoothness for affine functions

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

But,

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

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

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

But,

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

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.