Efficiency of Nash Equilibria

Maria Serna

Spring 2026

Efficiency at equilibrium

- We have analyzed existence of PNE.
- The players' goals can be different from those of the society.
- Fixing a social goal an optimal situation is possible.
- How good/bad are PNE with respect to this goal?does not achieve optimal travel time.

Efficiency at equilibrium

- We have analyzed existence of PNE.
- The players' goals can be different from those of the society.
- Fixing a social goal an optimal situation is possible.
- How good/bad are PNE with respect to this goal?does not achieve optimal travel time.
- How far are PNE for optimal social goal?

Efficiency at equilibrium

- We have analyzed existence of PNE.
- The players' goals can be different from those of the society.
- Fixing a social goal an optimal situation is possible.
- How good/bad are PNE with respect to this goal?does not achieve optimal travel time.
- How far are PNE 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.
- The players' goals can be different from those of the society.
- Fixing a social goal an optimal situation is possible.
- How good/bad are PNE with respect to this goal?does not achieve optimal travel time.
- How far are PNE 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.
- All that follows could be redefined replacing $PNE(\Gamma)$ by any other subset of strategy profiles of interest.

Social cost

- Consider a n-player game $\Gamma = (A_1, \dots, A_n, u_1, \dots, u_n)$.
- Let $A = A_1 \times \cdots \times A_n$.
- Let $PNE(\Gamma)$ be the set of PNE of Γ.
- All that follows could be redefined replacing $PNE(\Gamma)$ by any other subset of strategy profiles of interest.
- Let $\mathcal{C} : A \to \mathbb{R}$ be a social cost function.

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

Price of Anarchy/Stability

- 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 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.

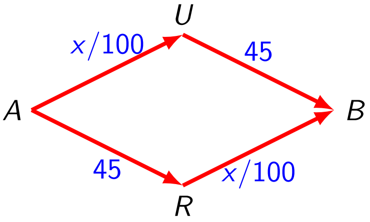
Price of Anarchy/Stability

- For 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.

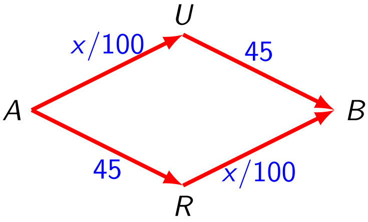
A network congestion game

- 4000 drivers drive from A to B on



A network congestion game

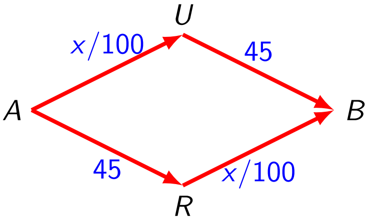
- 4000 drivers drive from A to B on



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

A network congestion game

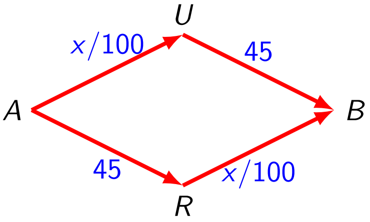
- 4000 drivers drive from A to B on



- 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.

A network congestion game

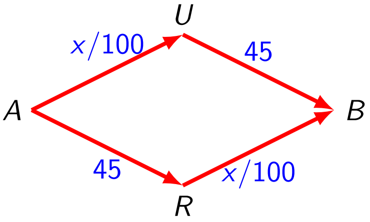
- 4000 drivers drive from A to B on



- 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

A network congestion game

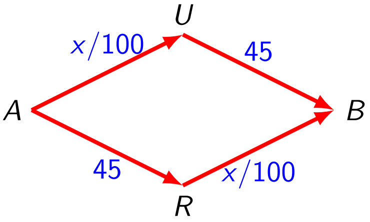
- 4000 drivers drive from A to B on



- 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 $A - R - B$.

A network congestion game

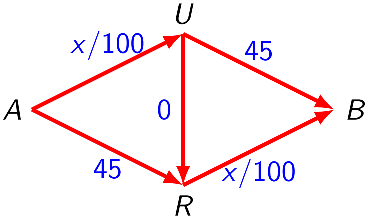
- 4000 drivers drive from A to B on



- 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 $A - R - B$.
- $PoA = PoS = 65/65 = 1$

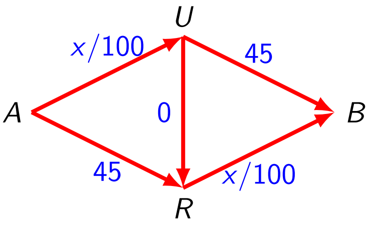
Braess' Network

- 4000 drivers drive from A to B on



Braess' Network

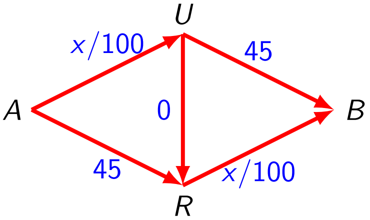
- 4000 drivers drive from A to B on



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

Braess' Network

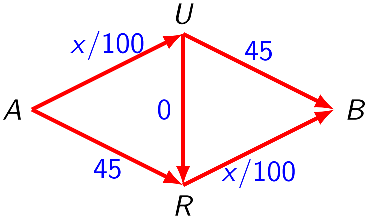
- 4000 drivers drive from A to B on



- 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

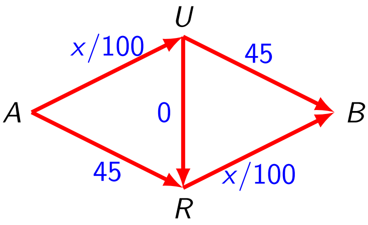
- 4000 drivers drive from A to B on



- 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

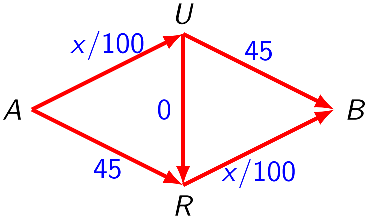
- 4000 drivers drive from A to B on



- 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.
- $PoA = PoS = 80/65 = 16/13$

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

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

- 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 following better response dynamic

- Start with an arbitrary state.
- A node chooses a better strategy, one that improves its own payoff, given the current choices of the others

Load Balancing game: PNE?

Consider the following better response dynamic

- Start with an arbitrary state.
- A node chooses a better strategy, one that improves 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 following better response dynamic

- Start with an arbitrary state.
- A node chooses a better strategy, one that improves 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, a function that increases at each step.

Load Balancing game: PNE?

BR-inspired-algorithm analysis

- Order the servers with decreasing load (i.e., the decreasing response time):

$$L_1 \geq L_2 \geq \cdots \geq L_m.$$

- Player i moves from server j to k if $L_k + p_i < L_j$.

Load Balancing game: PNE?

BR-inspired-algorithm analysis

- Order the servers with decreasing load (i.e., the decreasing response time):

$$L_1 \geq L_2 \geq \cdots \geq L_m.$$

- Player i moves from server j to k if $L_k + p_i < L_j$.
- Reorder the servers by decreasing load and repeat the process until no job can move.

Load Balancing game: PNE?

BR-inspired-algorithm analysis

- Order the servers with decreasing load (i.e., the decreasing response time):
$$L_1 \geq L_2 \geq \cdots \geq L_m.$$
- Player i moves from server j to k if $L_k + p_i < L_j$.
- Reorder the servers by decreasing load and repeat the process until no job can move.
- Each step of the BR algorithm defines a sorted sequence of loads.

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
  - Player i moves from server j to k if $L_k + p_i < L_j$.
  - Besides $L_j - p_i < L_j$

Load Balancing game: PNE?

- Does the algorithm converge?
- There are a finite number of (possibly exponential) assignments of jobs to servers.
- At each step
  - Player i moves from server j to k if $L_k + p_i < L_j$.
  - Besides $L_j - p_i < L_j$

  At least these two terms of the new sorted load sequence are smaller than L_j

  The new ordered sequence is lexicographically smaller than the previous one

Load Balancing game: PNE?

- Does the algorithm converge?
- There are a finite number of (possibly exponential) assignments of jobs to servers.
- At each step
  - Player i moves from server j to k if $L_k + p_i < L_j$.
  - Besides $L_j - p_i < L_j$

  At least these two terms of the new sorted load sequence are smaller than L_j

  The new ordered sequence is lexicographically smaller than the previous one
- 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
  - Player i moves from server j to k if $L_k + p_i < L_j$.
  - Besides $L_j - p_i < L_j$

  At least these two terms of the new sorted load sequence are smaller than L_j

  The new ordered sequence is lexicographically smaller than the previous one
- 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.
  There is a PNE with optimal cost.

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.
  There is a PNE with optimal cost.
- 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 $\sum_k L_k/m \leq (\sum_\ell p_\ell)/m$.

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 $\sum_k L_k/m \leq (\sum_\ell p_\ell)/m$.
- 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 $\sum_k L_k/m \leq (\sum_\ell p_\ell)/m$.
- 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 $\sum_k L_k/m \leq (\sum_\ell p_\ell)/m$.
- We get

$$C(s) = L_j \leq (\sum_k L_k)/m + p_i \leq (\sum_\ell p_\ell)/m + p_i$$
$$\leq C(s') + C(s').$$