Topics in Algorithmic Game Theory and Economics

Pieter Kleer

Max Planck Institute for Informatics (D1)
Saarland Informatics Campus

November 18, 2020

Lecture 2
Congestion Games I - Computation of PNE

Introduction

Congestion games can be used to model, e.g.,

Introduction

Congestion games can be used to model, e.g.,

- Traffic/routing games,
- Scheduling games,
- Broadcast games,
- Cost-sharing games.

Introduction

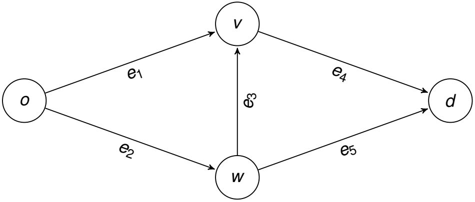
Congestion games can be used to model, e.g.,

- Traffic/routing games,
- Scheduling games,
- Broadcast games,
- Cost-sharing games.

Studied extensively in the last twenty years in the area of AGT.

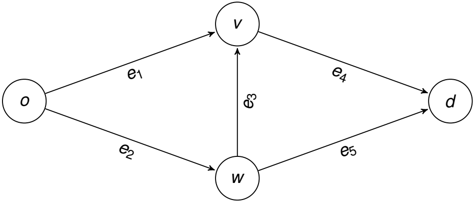
Atomic selfish routing game (example)

Given is directed graph $G = (V, E)$ with origin o and destination d.



Atomic selfish routing game (example)

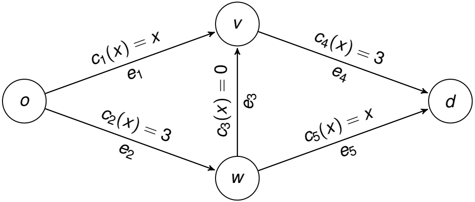
Given is directed graph $G = (V, E)$ with origin o and destination d.



- Symmetric strategy set of players in N are o, d-paths $\mathcal{P}$ in G.

Atomic selfish routing game (example)

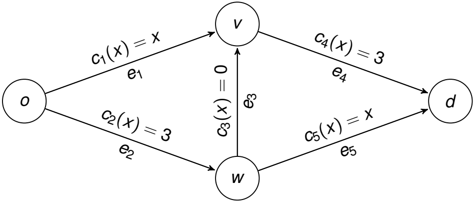
Given is directed graph $G = (V, E)$ with origin o and destination d.



- Symmetric strategy set of players in N are o, d-paths $\mathcal{P}$ in G.
- Arcs $e \in E$ have cost functions $c_e : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$.

Atomic selfish routing game (example)

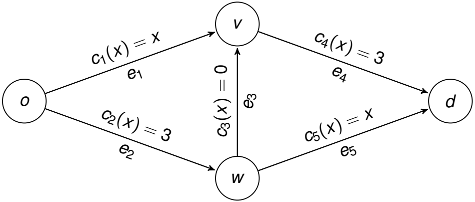
Given is directed graph $G = (V, E)$ with origin o and destination d.



- Symmetric strategy set of players in N are o, d-paths $\mathcal{P}$ in G.
- Arcs $e \in E$ have cost functions $c_e : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$.
  - We often write $c_i(x)$ instead of $c_{e_i}(x)$ for sake of readability.

Atomic selfish routing game (example)

Given is directed graph $G = (V, E)$ with origin o and destination d.

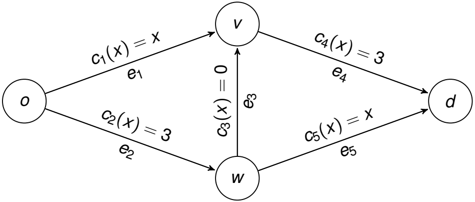


- Symmetric strategy set of players in N are o, d-paths $\mathcal{P}$ in G.
- Arcs $e \in E$ have cost functions $c_e : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$.
  - We often write $c_i(x)$ instead of $c_{e_i}(x)$ for sake of readability.

Players need to route one unsplittable unit of flow from o to d.

Atomic selfish routing game (example)

Given is directed graph $G = (V, E)$ with origin o and destination d.

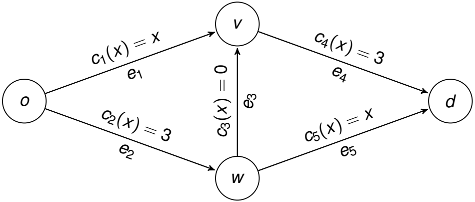


- Symmetric strategy set of players in N are o, d-paths $\mathcal{P}$ in G.
- Arcs $e \in E$ have cost functions $c_e : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$.
  - We often write $c_i(x)$ instead of $c_{e_i}(x)$ for sake of readability.

Players need to route one unsplittable unit of flow from o to d.
Goal is to choose path with cost as small as possible.

Atomic selfish routing game (example)

Given is directed graph $G = (V, E)$ with origin o and destination d.



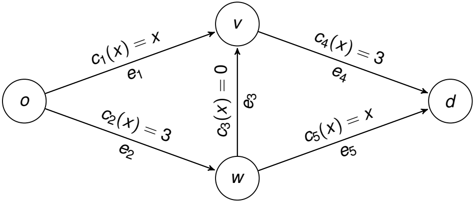
- Symmetric strategy set of players in N are o, d-paths $\mathcal{P}$ in G.
- Arcs $e \in E$ have cost functions $c_e : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$.
  - We often write $c_i(x)$ instead of $c_{e_i}(x)$ for sake of readability.

Players need to route one unsplittable unit of flow from o to d.
Goal is to choose path with cost as small as possible.

For strategy profile $s = (s_1, s_2, \ldots, s_n) \in \mathcal{P}^n$,

Atomic selfish routing game (example)

Given is directed graph $G = (V, E)$ with origin o and destination d.



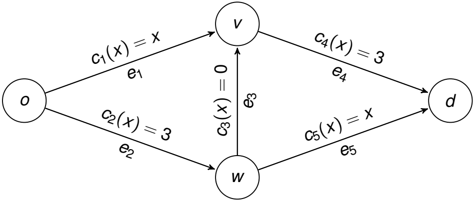
- Symmetric strategy set of players in N are o, d-paths $\mathcal{P}$ in G.
- Arcs $e \in E$ have cost functions $c_e : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$.
  - We often write $c_i(x)$ instead of $c_{e_i}(x)$ for sake of readability.

Players need to route one unsplittable unit of flow from o to d.
Goal is to choose path with cost as small as possible.

For strategy profile $s = (s_1, s_2, \ldots, s_n) \in \mathcal{P}^n$, with $x_e = x_e(s)$ number of players using $e \in E$,

Atomic selfish routing game (example)

Given is directed graph $G = (V, E)$ with origin o and destination d.



- Symmetric strategy set of players in N are o, d-paths $\mathcal{P}$ in G.
- Arcs $e \in E$ have cost functions $c_e : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$.
  - We often write $c_i(x)$ instead of $c_{e_i}(x)$ for sake of readability.

Players need to route one unsplittable unit of flow from o to d.
Goal is to choose path with cost as small as possible.

For strategy profile $s = (s_1, s_2, \ldots, s_n) \in \mathcal{P}^n$, with $x_e = x_e(s)$ number of players using $e \in E$,

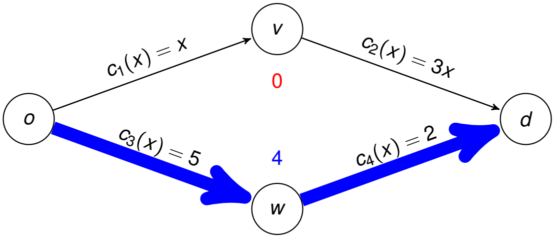
$$C_i(s) = \sum_{e \in s_i} c_e(x_e).$$

Atomic selfish routing game (cont'd)

Suppose we have $n = 4$ players and edges $E = \{e_1, \ldots, e_4\}$.

- Remember that player places one unit of flow on a path.
- Cost of player i in profile s is given by (with $s_i \in \{T, B\} = \mathcal{P}$)

$$C_i(s) = \sum_{e \in s_i} c_e(x_e).$$



$s = (B, B, B, B)$

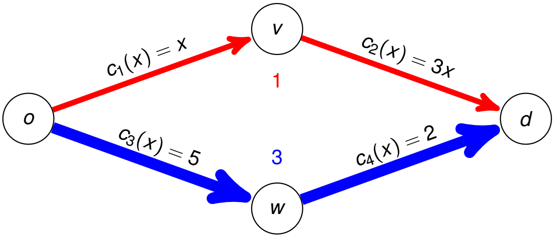
$$C_1(s) = 5 + 2 = 7$$
$$C_2(s) = 5 + 2 = 7$$
$$C_3(s) = 5 + 2 = 7$$
$$C_4(s) = 5 + 2 = 7$$

Atomic selfish routing game (cont'd)

Suppose we have $n = 4$ players and edges $E = \{e_1, \dots, e_4\}$.

- Remember that player places one unit of flow on a path.
- Cost of player i in profile s is given by (with $s_i \in \{T, B\} = \mathcal{P}$)

$$C_i(s) = \sum_{e \in s_i} c_e(x_e).$$



$s = (T, B, B, B)$

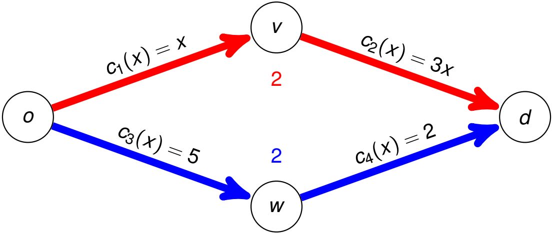
$$C_1(s) = 1 + 3 \cdot 1 = 4$$
$$C_2(s) = 5 + 2 = 7$$
$$C_3(s) = 5 + 2 = 7$$
$$C_4(s) = 5 + 2 = 7$$

Atomic selfish routing game (cont'd)

Suppose we have $n = 4$ players and edges $E = \{e_1, \dots, e_4\}$.

- Remember that player places one unit of flow on a path.
- Cost of player i in profile s is given by (with $s_i \in \{T, B\} = \mathcal{P}$)

$$C_i(s) = \sum_{e \in s_i} c_e(x_e).$$



$s = (T, T, B, B)$

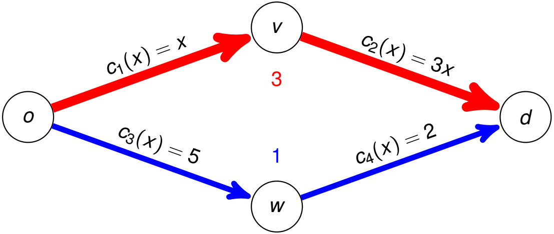
$$C_1(s) = 2 + 3 \cdot 2 = 8$$
$$C_2(s) = 2 + 3 \cdot 2 = 8$$
$$C_3(s) = 5 + 2 = 7$$
$$C_4(s) = 5 + 2 = 7$$

Atomic selfish routing game (cont'd)

Suppose we have $n = 4$ players and edges $E = \{e_1, \dots, e_4\}$.

- Remember that player places one unit of flow on a path.
- Cost of player i in profile s is given by (with $s_i \in \{T, B\} = \mathcal{P}$)

$$C_i(s) = \sum_{e \in s_i} c_e(x_e).$$



$s = (T, T, T, B)$

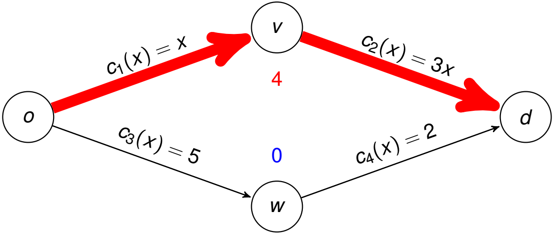
$$C_1(s) = 3 + 3 \cdot 3 = 12$$
$$C_2(s) = 3 + 3 \cdot 3 = 12$$
$$C_3(s) = 3 + 3 \cdot 3 = 12$$
$$C_4(s) = 5 + 2 = 7$$

Atomic selfish routing game (cont'd)

Suppose we have $n = 4$ players and edges $E = \{e_1, \ldots, e_4\}$.

- Remember that player places one unit of flow on a path.
- Cost of player i in profile s is given by (with $s_i \in \{T, B\} = \mathcal{P}$)

$$C_i(s) = \sum_{e \in s_i} c_e(x_e).$$



$s = (T, T, T, T)$

$$C_1(s) = 4 + 3 \cdot 4 = 16$$
$$C_2(s) = 4 + 3 \cdot 4 = 16$$
$$C_3(s) = 4 + 3 \cdot 4 = 16$$
$$C_3(s) = 4 + 3 \cdot 4 = 16$$

Congestion games

(Atomic) congestion game $\Gamma = (N, E, (\mathcal{S}_i)_{i \in N}, (c_e)_{e \in E})$:

Congestion games

(Atomic) congestion game $\Gamma = (N, E, (\mathcal{S}_i)_{i \in N}, (c_e)_{e \in E})$:

- Set of players $N = \{1, \ldots, n\}$.

Congestion games

(Atomic) congestion game $\Gamma = (N, E, (\mathcal{S}_i)_{i \in N}, (c_e)_{e \in E})$:

- Set of players $N = \{1, \ldots, n\}$.
- Set of resources $E = \{e_1, \ldots, e_m\}$.

Congestion games

(Atomic) congestion game $\Gamma = (N, E, (\mathcal{S}_i)_{i \in N}, (c_e)_{e \in E})$:

- Set of players $N = \{1, \ldots, n\}$.
- Set of resources $E = \{e_1, \ldots, e_m\}$.
- Strategy set $\mathcal{S}_i \subseteq 2^E = \{X : X \subseteq E\}$ for all $i \in N$.

Congestion games

(Atomic) congestion game $\Gamma = (N, E, (\mathcal{S}_i)_{i \in N}, (c_e)_{e \in E})$:

- Set of players $N = \{1, \ldots, n\}$.
- Set of resources $E = \{e_1, \ldots, e_m\}$.
- Strategy set $\mathcal{S}_i \subseteq 2^E = \{X : X \subseteq E\}$ for all $i \in N$.
    - (o, d)*-paths in directed graph.*

Congestion games

(Atomic) congestion game $\Gamma = (N, E, (\mathcal{S}_i)_{i \in N}, (c_e)_{e \in E})$:

- Set of players $N = \{1, \ldots, n\}$.
- Set of resources $E = \{e_1, \ldots, e_m\}$.
- Strategy set $\mathcal{S}_i \subseteq 2^E = \{X : X \subseteq E\}$ for all $i \in N$.
  - *(o, d)-paths in directed graph.*
- Cost function $c_e : \mathbb{R}_{\geq 0} \to \mathbb{R}$ for $e \in E$.

Congestion games

(Atomic) congestion game $\Gamma = (N, E, (\mathcal{S}_i)_{i \in N}, (c_e)_{e \in E})$:

- Set of players $N = \{1, \ldots, n\}$.
- Set of resources $E = \{e_1, \ldots, e_m\}$.
- Strategy set $\mathcal{S}_i \subseteq 2^E = \{X : X \subseteq E\}$ for all $i \in N$.
  - *(o, d)-paths in directed graph.*
- Cost function $c_e : \mathbb{R}_{\geq 0} \to \mathbb{R}$ for $e \in E$.
  - *Although the word 'congestion' hints at these functions being non-decreasing, this is not required.*

Congestion games

(Atomic) congestion game $\Gamma = (N, E, (\mathcal{S}_i)_{i \in N}, (c_e)_{e \in E})$:

- Set of players $N = \{1, \dots, n\}$.
- Set of resources $E = \{e_1, \dots, e_m\}$.
- Strategy set $\mathcal{S}_i \subseteq 2^E = \{X : X \subseteq E\}$ for all $i \in N$.
  - *(o, d)-paths in directed graph.*
- Cost function $c_e : \mathbb{R}_{\geq 0} \to \mathbb{R}$ for $e \in E$.
  - *Although the word 'congestion' hints at these functions being non-decreasing, this is not required.*

Player places one unit of unsplittable load on a strategy with goal of minimizing her cost.

Congestion games

(Atomic) congestion game $\Gamma = (N, E, (\mathcal{S}_i)_{i \in N}, (c_e)_{e \in E})$:

- Set of players $N = \{1, \ldots, n\}$.
- Set of resources $E = \{e_1, \ldots, e_m\}$.
- Strategy set $\mathcal{S}_i \subseteq 2^E = \{X : X \subseteq E\}$ for all $i \in N$.
    - *(o, d)-paths in directed graph.*
- Cost function $c_e : \mathbb{R}_{\geq 0} \to \mathbb{R}$ for $e \in E$.
    - *Although the word 'congestion' hints at these functions being non-decreasing, this is not required.*

Player places one unit of unsplittable load on a strategy with goal of minimizing her cost.

For strategy profile $s = (s_1, s_2, \ldots, s_n) \in S_1 \times S_2 \times \cdots \times S_n = \times_i \mathcal{S}_i$,

$$C_i(s) = \sum_{e \in s_i} c_e(x_e),$$

where $x_e = x_e(s)$ is the number of players using $e \in E$, i.e., the load.

Congestion games

(Atomic) congestion game $\Gamma = (N, E, (\mathcal{S}_i)_{i \in N}, (c_e)_{e \in E})$:

- Set of players $N = \{1, \ldots, n\}$.
- Set of resources $E = \{e_1, \ldots, e_m\}$.
- Strategy set $\mathcal{S}_i \subseteq 2^E = \{X : X \subseteq E\}$ for all $i \in N$.
    - *(o, d)-paths in directed graph.*
- Cost function $c_e : \mathbb{R}_{\geq 0} \to \mathbb{R}$ for $e \in E$.
    - *Although the word 'congestion' hints at these functions being non-decreasing, this is not required.*

Player places one unit of unsplittable load on a strategy with goal of minimizing her cost.

For strategy profile $s = (s_1, s_2, \ldots, s_n) \in S_1 \times S_2 \times \cdots \times S_n = \times_i \mathcal{S}_i$,

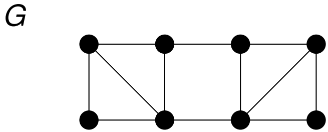
$$C_i(s) = \sum_{e \in s_i} c_e(x_e),$$

where $x_e = x_e(s)$ is the number of players using $e \in E$, i.e., the load.

Broadcast game (example)

Given is undirected graph $G = (V, E)$.

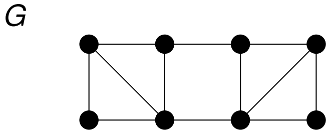
- Edges $e \in E$ are resources with cost function c_e.
- Players place one unit of unsplittable load on spanning tree of G.
    - Spanning trees are the strategies of the players.



Broadcast game (example)

Given is undirected graph $G = (V, E)$.

- Edges $e \in E$ are resources with cost function c_e.
- Players place one unit of unsplittable load on spanning tree of G.
    - Spanning trees are the strategies of the players.

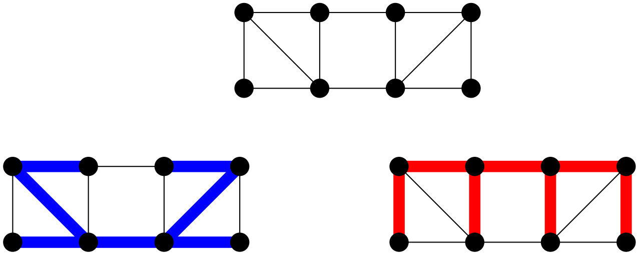


Broadcast game (example)

Given is undirected graph $G = (V, E)$.

- Edges $e \in E$ are resources with cost function c_e.
- Players place one unit of unsplittable load on spanning tree of G.
  - Spanning trees are the strategies of the players.

G



Example of base (graphic) matroid congestion game.

Pure Nash equilibrium

We will focus on pure Nash equilibria in congestion games.

Definition (Pure Nash equilibrium (PNE))

A strategy profile $s \in \times_i \mathcal{S}_i$ is a pure Nash equilibrium if for every $i \in N$,

$$C_i(s_1, \ldots, s_i, \ldots, s_n) \leq C_i(s_1, \ldots, s_i', \ldots, s_n)$$

for every $s_i' \in \mathcal{S}_i$.

Pure Nash equilibrium

We will focus on pure Nash equilibria in congestion games.

Definition (Pure Nash equilibrium (PNE))

A strategy profile $s \in \times_i \mathcal{S}_i$ is a pure Nash equilibrium if for every $i \in N$,

$$C_i(s_1, \ldots, s_i, \ldots, s_n) \leq C_i(s_1, \ldots, s_i', \ldots, s_n)$$

for every $s_i' \in \mathcal{S}_i$. *In short,* $C_i(s) \leq C_i(s_i', s_{-i})$.

Pure Nash equilibrium

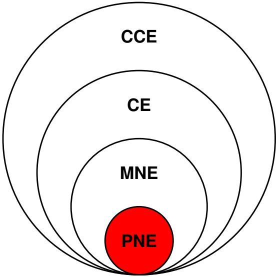
We will focus on pure Nash equilibria in congestion games.

Definition (Pure Nash equilibrium (PNE))

A strategy profile $s \in \times_i \mathcal{S}_i$ is a pure Nash equilibrium if for every $i \in N$,

$$C_i(s_1, \ldots, s_i, \ldots, s_n) \leq C_i(s_1, \ldots, s_i', \ldots, s_n)$$

for every $s_i' \in \mathcal{S}_i$. *In short,* $C_i(s) \leq C_i(s_i', s_{-i})$.



Why focus on PNE?

Pure Nash equilibrium

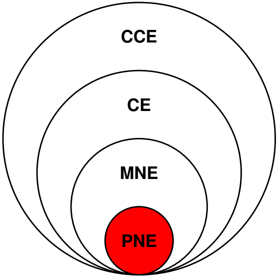
We will focus on pure Nash equilibria in congestion games.

Definition (Pure Nash equilibrium (PNE))

A strategy profile $s \in \times_i \mathcal{S}_i$ is a pure Nash equilibrium if for every $i \in N$,

$$C_i(s_1, \ldots, s_i, \ldots, s_n) \leq C_i(s_1, \ldots, s_i', \ldots, s_n)$$

for every $s_i' \in \mathcal{S}_i$. *In short,* $C_i(s) \leq C_i(s_i', s_{-i})$.



Why focus on PNE?
There always exists at least one!

Pure Nash equilibrium

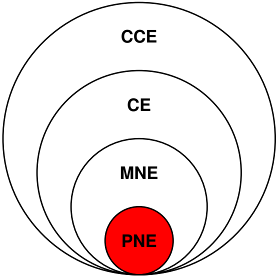
We will focus on pure Nash equilibria in congestion games.

Definition (Pure Nash equilibrium (PNE))

A strategy profile $s \in \times_i \mathcal{S}_i$ is a pure Nash equilibrium if for every $i \in N$,

$$C_i(s_1, \ldots, s_i, \ldots, s_n) \leq C_i(s_1, \ldots, s_i', \ldots, s_n)$$

for every $s_i' \in \mathcal{S}_i$. *In short,* $C_i(s) \leq C_i(s_i', s_{-i})$.



Why focus on PNE?
There always exists at least one!

Potential function method

Potential function method

Show existence of potential function $\Phi : \times_i \mathcal{S}_i \to \mathbb{R}$ tracking improvements in player costs.

That is, Φ has the property that if, in strategy profile $s = (s_1, \ldots, s_n)$,

Potential function method

Show existence of potential function $\Phi : \times_i \mathcal{S}_i \to \mathbb{R}$ tracking improvements in player costs.

That is, Φ has the property that if, in strategy profile $s = (s_1, \ldots, s_n)$,

- Player i has improving move by switching to $s'_i \in \mathcal{S}_i$, i.e.,

$$C_i(s'_i, s_{-i}) < C_i(s).$$

Potential function method

Show existence of potential function $\Phi : \times_i \mathcal{S}_i \to \mathbb{R}$ tracking improvements in player costs.

That is, Φ has the property that if, in strategy profile $s = (s_1, \ldots, s_n)$,

- Player i has improving move by switching to $s_i' \in \mathcal{S}_i$, i.e.,

$$C_i(s_i', s_{-i}) < C_i(s).$$

- Then also

$$\Phi(s_i', s_{-i}) < \Phi(s).$$

Potential function method

Show existence of potential function $\Phi : \times_i \mathcal{S}_i \to \mathbb{R}$ tracking improvements in player costs.

That is, Φ has the property that if, in strategy profile $s = (s_1, \ldots, s_n)$,

- Player i has improving move by switching to $s_i' \in \mathcal{S}_i$, i.e.,

$$C_i(s_i', s_{-i}) < C_i(s).$$

- Then also

$$\Phi(s_i', s_{-i}) < \Phi(s).$$

ALGORITHM 4: Better response dynamics

Input : Strategy profile $s^0 \in \times_i \mathcal{S}_i$.
Output: Pure Nash equilibrium s^*.

$k = 0$.
while s^k is not a pure Nash equilibrium **do**
  Select player $i \in N$ and $s_i' \in \mathcal{S}_i$ such that $C_i(s_i', s_{-i}) < C_i(s)$.
  $s^{k+1} \leftarrow (s_i', s_{-i}^k)$.
  $k \leftarrow k + 1$.
end
return $s^* \leftarrow s^k$

Better response dynamics always terminate (converge) in finite number of steps, given the existence of the function Φ.

Better response dynamics always terminate (converge) in finite number of steps, given the existence of the function Φ.

Why?

Better response dynamics always terminate (converge) in finite number of steps, given the existence of the function Φ.

Why?

- If player i makes improving move in step k, then $\Phi(s^{k+1}) < \Phi(s^k)$.

Better response dynamics always terminate (converge) in finite number of steps, given the existence of the function Φ.

Why?

- If player i makes improving move in step k, then $\Phi(s^{k+1}) < \Phi(s^k)$.
  - This means

$$\cdots < \Phi(s^{k+1}) < \Phi(s^k) < \Phi(s^{k-1}) < \cdots < \Phi(s^1) < \Phi(s^0).$$

Better response dynamics always terminate (converge) in finite number of steps, given the existence of the function Φ.

Why?

- If player i makes improving move in step k, then $\Phi(s^{k+1}) < \Phi(s^k)$.
    - This means

$$\cdots < \Phi(s^{k+1}) < \Phi(s^k) < \Phi(s^{k-1}) < \cdots < \Phi(s^1) < \Phi(s^0).$$

- **There are only finitely many strategy profiles.**

Better response dynamics always terminate (converge) in finite number of steps, given the existence of the function Φ.

Why?

- If player i makes improving move in step k, then $\Phi(s^{k+1}) < \Phi(s^k)$.
  - This means

$$\cdots < \Phi(s^{k+1}) < \Phi(s^k) < \Phi(s^{k-1}) < \cdots < \Phi(s^1) < \Phi(s^0).$$

- **There are only finitely many strategy profiles.**
  - Remember that we assume that $\mathcal{S}_i$ is finite for every $i \in N$.

Better response dynamics always terminate (converge) in finite number of steps, given the existence of the function Φ.

Why?

- If player i makes improving move in step k, then $\Phi(s^{k+1}) < \Phi(s^k)$.
  - This means

$$\cdots < \Phi(s^{k+1}) < \Phi(s^k) < \Phi(s^{k-1}) < \cdots < \Phi(s^1) < \Phi(s^0).$$

- **There are only finitely many strategy profiles.**
  - Remember that we assume that $\mathcal{S}_i$ is finite for every $i \in N$.

Theorem (Rosenthal, 1973)

Every (finite) congestion game possesses a pure Nash equilibrium. It can be computed by better response dynamics.

Better response dynamics always terminate (converge) in finite number of steps, given the existence of the function Φ.

Why?

- If player i makes improving move in step k, then $\Phi(s^{k+1}) < \Phi(s^k)$.
  - This means

$$\cdots < \Phi(s^{k+1}) < \Phi(s^k) < \Phi(s^{k-1}) < \cdots < \Phi(s^1) < \Phi(s^0).$$

- **There are only finitely many strategy profiles.**
  - Remember that we assume that $\mathcal{S}_i$ is finite for every $i \in N$.

Theorem (Rosenthal, 1973)

Every (finite) congestion game possesses a pure Nash equilibrium. It can be computed by better response dynamics.

What is the potential function Φ?

Rosenthal's potential

The Rosenthal (potential) function $\Phi : \times_i \mathcal{S}_i \to \mathbb{R}$ is given by

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k).$$

Rosenthal's potential

The Rosenthal (potential) function $\Phi : \times_i \mathcal{S}_i \to \mathbb{R}$ is given by

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k).$$

Remember that $C_i(s) = \sum_{e \in s_i} c_e(x_e)$.

- $x_e = x_e(s)$ total number of players using resource e in s.

Rosenthal's potential

The Rosenthal (potential) function $\Phi : \times_i \mathcal{S}_i \to \mathbb{R}$ is given by

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k).$$

Remember that $C_i(s) = \sum_{e \in s_i} c_e(x_e)$.

- $x_e = x_e(s)$ total number of players using resource e in s.

Lemma (Rosenthal's potential)

Rosenthal's potential satisfies, for every $i \in N$ and $s_i' \in \mathcal{S}_i$,

$$C_i(s) - C_i(s_i', s_{-i}) = \Phi(s) - \Phi(s_i', s_{-i}).$$

Rosenthal's potential

The Rosenthal (potential) function $\Phi : \times_i \mathcal{S}_i \to \mathbb{R}$ is given by

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k).$$

Remember that $C_i(s) = \sum_{e \in s_i} c_e(x_e)$.

- $x_e = x_e(s)$ total number of players using resource e in s.

Lemma (Rosenthal's potential)

Rosenthal's potential satisfies, for every $i \in N$ and $s_i' \in \mathcal{S}_i$,

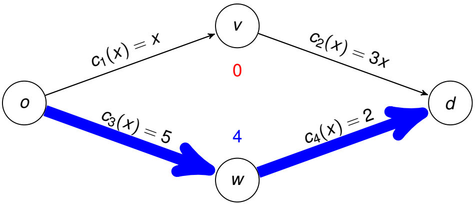
$$C_i(s) - C_i(s_i', s_{-i}) = \Phi(s) - \Phi(s_i', s_{-i}).$$

- *Proof (sketch) on Slide 12 for symmetric singleton games.*
- Exercise: Generalize the proof to general congestion games.

Rosenthal's potential (example)

Remember, for strategy profile s,

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k).$$



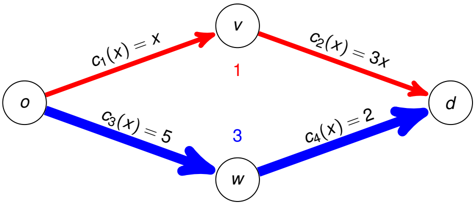
$$\begin{aligned} \Phi(s) &= 0 \\ &+ 0 \\ &+ [c_3(1) + c_3(2) + c_3(3) + c_3(4)] \\ &+ [c_4(1) + c_4(2) + c_4(3) + c_4(4)] \\ &= 28 \end{aligned}$$

$C_1(s) = 5 + 2 = 7$
$C_2(s) = 5 + 2 = 7$
$C_3(s) = 5 + 2 = 7$
$C_4(s) = 5 + 2 = 7$

Rosenthal's potential (example)

Remember, for strategy profile s,

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k).$$



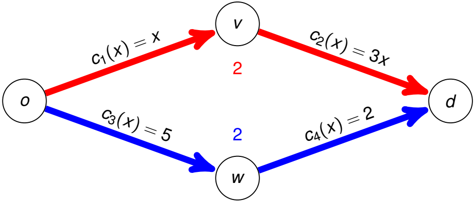
$$C_1(s) = 1 + 3 \cdot 1 = 4$$
$$C_2(s) = 5 + 2 = 7$$
$$C_3(s) = 5 + 2 = 7$$
$$C_4(s) = 5 + 2 = 7$$

$$\begin{aligned} \Phi(s) &= [c_1(1)] \\ &+ [c_2(1)] \\ &+ [c_3(1) + c_3(2) + c_3(3)] \\ &+ [c_4(1) + c_4(2) + c_4(3)] \\ &= 25 \end{aligned}$$

Rosenthal's potential (example)

Remember, for strategy profile s,

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k).$$



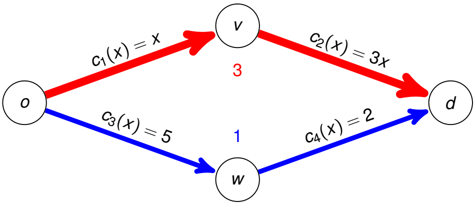
$$\begin{aligned}\Phi(s) &= [c_1(1) + c_1(2)] \\ &+ [c_2(1) + c_2(2)] \\ &+ [c_3(1) + c_3(2)] \\ &+ [c_4(1) + c_4(2)] \\ &= 26\end{aligned}$$

$$C_1(s) = 2 + 3 \cdot 2 = 8$$
$$C_2(s) = 2 + 3 \cdot 2 = 8$$
$$C_3(s) = 5 + 2 = 7$$
$$C_4(s) = 5 + 2 = 7$$

Rosenthal's potential (example)

Remember, for strategy profile s,

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k).$$



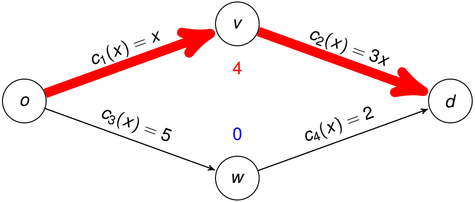
$$\begin{aligned} \Phi(s) &= [c_1(1) + c_1(2) + c_1(3)] \\ &\quad + [c_2(1) + c_2(2) + c_2(3)] \\ &\quad + [c_3(1)] \\ &\quad + [c_4(1)] \\ &= 31 \end{aligned}$$

$$C_1(s) = 3 + 3 \cdot 3 = 12$$
$$C_2(s) = 3 + 3 \cdot 3 = 12$$
$$C_3(s) = 3 + 3 \cdot 3 = 12$$
$$C_4(s) = 5 + 2 = 7$$

Rosenthal's potential (example)

Remember, for strategy profile s,

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k).$$



$$\begin{aligned}
\Phi(s) &= [c_1(1) + c_1(2) + c_1(3) + c_1(4)] \\
&\quad + [c_2(1) + c_2(2) + c_2(3) + c_2(4)] \\
&\quad + 0 \\
&\quad + 0 \\
&= 40
\end{aligned}$$

$$\begin{aligned}
C_1(s) &= 4 + 3 \cdot 4 = 16 \\
C_2(s) &= 4 + 3 \cdot 4 = 16 \\
C_3(s) &= 4 + 3 \cdot 4 = 16 \\
C_3(s) &= 4 + 3 \cdot 4 = 16
\end{aligned}$$

Symmetric singleton game $\Gamma = (N, E, (\mathcal{S}_i), (c_e))$ given by

$$\mathcal{S}_i = \{\{e_1\}, \{e_2\}, \ldots, \{e_m\}\}$$

Symmetric singleton game $\Gamma = (N, E, (\mathcal{S}_i), (c_e))$ given by

$$\mathcal{S}_i = \{\{e_1\}, \{e_2\}, \ldots, \{e_m\}\}$$

- That is, every player has to choose one resource from the set E.

Symmetric singleton game $\Gamma = (N, E, (\mathcal{S}_i), (c_e))$ given by

$$\mathcal{S}_i = \{\{e_1\}, \{e_2\}, \ldots, \{e_m\}\}$$

- That is, every player has to choose <u>one</u> resource from the set E.

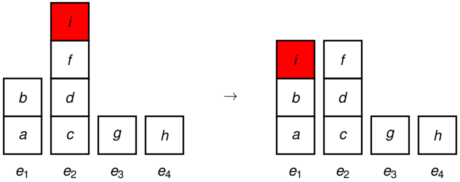
Remember $\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k)$ ***and*** $C_i(s) = \sum_{e \in s_i} c_e(x_e)$.

Symmetric singleton game $\Gamma = (N, E, (\mathcal{S}_i), (c_e))$ given by

$$\mathcal{S}_i = \{\{e_1\}, \{e_2\}, \ldots, \{e_m\}\}$$

- That is, every player has to choose <u>one</u> resource from the set E.

Remember $\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k)$ ***and*** $C_i(s) = \sum_{e \in s_i} c_e(x_e)$***.***

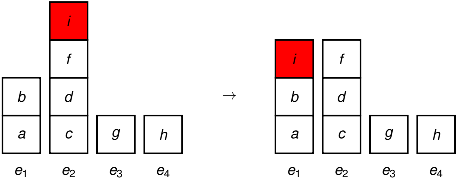


Symmetric singleton game $\Gamma = (N, E, (\mathcal{S}_i), (c_e))$ given by

$$\mathcal{S}_i = \{\{e_1\}, \{e_2\}, \dots, \{e_m\}\}$$

- That is, every player has to choose <u>one</u> resource from the set E.

Remember $\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k)$ ***and*** $C_i(s) = \sum_{e \in s_i} c_e(x_e)$***.***



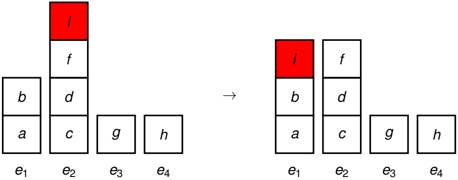
$$
\begin{aligned}
C_i(s) - C_i(s_i', s_{-i}) &= c_2(4) - c_1(3) \\
&= c_2(x_2(s)) - c_1(x_1(s) + 1)
\end{aligned}
$$

Symmetric singleton game $\Gamma = (N, E, (\mathcal{S}_i), (c_e))$ given by

$$\mathcal{S}_i = \{\{e_1\}, \{e_2\}, \dots, \{e_m\}\}$$

- That is, every player has to choose <u>one</u> resource from the set E.

Remember $\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k)$ ***and*** $C_i(s) = \sum_{e \in s_i} c_e(x_e)$.



$$\begin{aligned} C_i(s) - C_i(s_i', s_{-i}) &= c_2(4) - c_1(3) \\ &= c_2(x_2(s)) - c_1(x_1(s) + 1) \end{aligned}$$

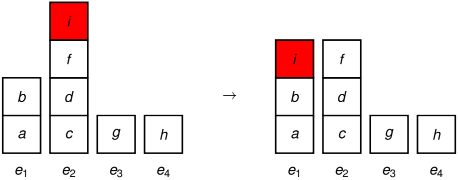
$$\begin{aligned} \Phi(s) - \Phi(s_i', s_{-i}) &= [c_1(1) + c_1(2)] + [c_2(1) + c_2(2) + c_2(3) + c_2(4)] \\ &+ c_3(1) + c_4(1) \\ &- [c_1(1) + c_1(2) + c_1(3)] - [c_2(1) + c_2(2) + c_2(3)] \\ &- c_3(1) - c_4(1) \end{aligned}$$

Symmetric singleton game $\Gamma = (N, E, (\mathcal{S}_i), (c_e))$ given by

$$\mathcal{S}_i = \{\{e_1\}, \{e_2\}, \dots, \{e_m\}\}$$

- That is, every player has to choose one resource from the set E.

Remember $\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k)$ ***and*** $C_i(s) = \sum_{e \in s_i} c_e(x_e)$.



$$\begin{aligned}
C_i(s) - C_i(s_i', s_{-i}) &= c_2(4) - c_1(3) \\
&= c_2(x_2(s)) - c_1(x_1(s) + 1)
\end{aligned}$$

$$\begin{aligned}
\Phi(s) - \Phi(s_i', s_{-i}) &= [c_1(1) + c_1(2)] + [c_2(1) + c_2(2) + c_2(3) + c_2(4)] \\
&+ c_3(1) + c_4(1) \\
&- [c_1(1) + c_1(2) + c_1(3)] - [c_2(1) + c_2(2) + c_2(3)] \\
&- c_3(1) - c_4(1) \\
&= c_2(4) - c_1(3) \\
&= c_2(x_2(s)) - c_1(x_1(s) + 1) \qquad \square
\end{aligned}$$

Brief overview

PNE always exists and can be computed by better response dynamics.

Brief overview

PNE always exists and can be computed by better response dynamics.

In fact, we showed that congestion games are exact potential games.

Brief overview

PNE always exists and can be computed by better response dynamics.

In fact, we showed that congestion games are exact potential games.

Definition (Exact potential game)

Finite game $\Gamma = (N, (\mathcal{S}_i), (C_i))$ is exact potential game if there exists function $\Phi : \times_i \mathcal{S}_i \to \mathbb{R}$ such that

$$C_i(s) - C_i(s'_i, s_{-i}) = \Phi(s) - \Phi(s'_i, s_{-i})$$

for every $i \in N$ and $s'_i \in \mathcal{S}_i$.

Brief overview

PNE always exists and can be computed by better response dynamics.

In fact, we showed that congestion games are exact potential games.

Definition (Exact potential game)

Finite game $\Gamma = (N, (\mathcal{S}_i), (C_i))$ is exact potential game if there exists function $\Phi : \times_i \mathcal{S}_i \to \mathbb{R}$ such that

$$C_i(s) - C_i(s_i', s_{-i}) = \Phi(s) - \Phi(s_i', s_{-i})$$

for every $i \in N$ and $s_i' \in \mathcal{S}_i$.

Theorem

The class of congestion games is 'isomorphic' to the class of exact potential games.

Algorithmic questions

Of interest to the computer scientist:

Algorithmic questions

Of interest to the computer scientist:

- Do better response dynamics converge in poly-time to PNE?

Algorithmic questions

Of interest to the computer scientist:

- Do better response dynamics converge in poly-time to PNE?
- If not, can we compute PNE in polynomial time by other means?

Algorithmic questions

Of interest to the computer scientist:

- Do better response dynamics converge in poly-time to PNE?
- If not, can we compute PNE in polynomial time by other means?

For both questions: In general NO, but in certain special cases YES.

Algorithmic questions

Of interest to the computer scientist:

- Do better response dynamics converge in poly-time to PNE?
- If not, can we compute PNE in polynomial time by other means?

For both questions: In general NO, but in certain special cases YES.

Polynomial in parameters needed to specify player costs in game.

- In general nk^n numbers are needed for this where $k = \max_i |\mathcal{S}_i|$.

Algorithmic questions

Of interest to the computer scientist:

- Do better response dynamics converge in poly-time to PNE?
- If not, can we compute PNE in polynomial time by other means?

For both questions: In general NO, but in certain special cases YES.

Polynomial in parameters needed to specify player costs in game.

- In general nk^n numbers are needed for this where $k = \max_i |\mathcal{S}_i|$.
- Many special cases can be represented more compactly.

Algorithmic questions

Of interest to the computer scientist:

- Do better response dynamics converge in poly-time to PNE?
- If not, can we compute PNE in polynomial time by other means?

For both questions: In general NO, but in certain special cases YES.

Polynomial in parameters needed to specify player costs in game.

- In general nk^n numbers are needed for this where $k = \max_i |\mathcal{S}_i|$.
- Many special cases can be represented more compactly.
  - For positive answers to the above questions, we usually get $\text{poly}(n, m, |c|)$-running time.

Algorithmic questions

Of interest to the computer scientist:

- Do better response dynamics converge in poly-time to PNE?
- If not, can we compute PNE in polynomial time by other means?

For both questions: In general NO, but in certain special cases YES.

Polynomial in parameters needed to specify player costs in game.

- In general nk^n numbers are needed for this where $k = \max_i |\mathcal{S}_i|$.
- Many special cases can be represented more compactly.
  - For positive answers to the above questions, we usually get $\mathrm{poly}(n, m, |c|)$-running time.

How to study computational complexity of PNE in congestion games?

Algorithmic questions

Of interest to the computer scientist:

- Do better response dynamics converge in poly-time to PNE?
- If not, can we compute PNE in polynomial time by other means?

For both questions: In general NO, but in certain special cases YES.

Polynomial in parameters needed to specify player costs in game.

- In general nk^n numbers are needed for this where $k = \max_i |\mathcal{S}_i|$.
- Many special cases can be represented more compactly.
  - For positive answers to the above questions, we usually get $\text{poly}(n, m, |c|)$-running time.

How to study computational complexity of PNE in congestion games?
*Interpret it as **local search problem** w.r.t. Rosenthal's potential.*

Remark (for Homework 1)

In all statements on previous slides, we can replace ‘better’ by ‘best’.

Remark (for Homework 1)

In all statements on previous slides, we can replace ‘better’ by ‘best’.

Best response dynamics

In better response dynamics algorithm, always choose strategy yielding best improvement in cost.

Remark (for Homework 1)

In all statements on previous slides, we can replace 'better' by 'best'.

Best response dynamics

In better response dynamics algorithm, always choose strategy yielding best improvement in cost.

ALGORITHM 7: Best response dynamics

Input : Strategy profile $s^0 \in \times_i \mathcal{S}_i$.
Output: Pure Nash equilibrium s^*.

$k = 0$.
while s^k is not a pure Nash equilibrium **do**
    Select player $i \in N$ and $s_i' \in \mathcal{S}_i$ such that

$$C_i(s_i', s_{-i}) = \min_{t_i \in \mathcal{S}_i} C_i(t_i, s_{-i}).$$

    $s^{k+1} \leftarrow (s_i', s_{-i}^k)$.
    $k \leftarrow k + 1$.
end
return $s^* \leftarrow s^k$

Some positive results to algorithmic questions

1. Special cases where response dynamics converge quickly:
   - Better response dynamics in singleton congestion games.
   - *Best response dynamics in base matroid congestion games.*
     - Homework 1.

Some positive results to algorithmic questions

1. Special cases where response dynamics converge quickly:
   - Better response dynamics in singleton congestion games.
   - *Best response dynamics in base matroid congestion games.*
     - Homework 1.
2. Special case where PNE can be computed by other means:
   - Symmetric network congestion games.

Singleton congestion games

Definition

A singleton congestion game $\Gamma = (N, E, (\mathcal{S}_i), (c_e))$ has the property that $\mathcal{S}_i \subseteq \{\{e_1, \}, \{e_2\}, \ldots, \{e_m\}\}$, i.e., every possible strategy consists of a single resource.

Singleton congestion games

Definition

A singleton congestion game $\Gamma = (N, E, (\mathcal{S}_i), (c_e))$ has the property that $\mathcal{S}_i \subseteq \{\{e_1,\}, \{e_2\}, \ldots, \{e_m\}\}$, i.e., every possible strategy consists of a single resource.

Theorem (Ieong et al., 2005)

For singleton congestion games, better response dynamics (BRD) terminate in at most $n^2 m$ steps (with n #players and m #resources).

- *Proof on next slide.*

Singleton congestion games

Definition

A singleton congestion game $\Gamma = (N, E, (\mathcal{S}_i), (c_e))$ has the property that $\mathcal{S}_i \subseteq \{\{e_1,\}, \{e_2\}, \ldots, \{e_m\}\}$, i.e., every possible strategy consists of a single resource.

Theorem (Ieong et al., 2005)

For singleton congestion games, better response dynamics (BRD) terminate in at most $n^2 m$ steps (with n #players and m #resources).

- *Proof on next slide. Remember that* $\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k)$.

Singleton congestion games

Definition

A singleton congestion game $\Gamma = (N, E, (\mathcal{S}_i), (c_e))$ has the property that $\mathcal{S}_i \subseteq \{\{e_1,\}, \{e_2\}, \ldots, \{e_m\}\}$, i.e., every possible strategy consists of a single resource.

Theorem (Ieong et al., 2005)

For singleton congestion games, better response dynamics (BRD) terminate in at most $n^2 m$ steps (with n #players and m #resources).

- *Proof on next slide. Remember that $\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k)$.*

Lemma

If cost functions (c_e) are integer-valued,

Singleton congestion games

Definition

A singleton congestion game $\Gamma = (N, E, (\mathcal{S}_i), (c_e))$ has the property that $\mathcal{S}_i \subseteq \{\{e_1,\}, \{e_2\}, \ldots, \{e_m\}\}$, i.e., every possible strategy consists of a single resource.

Theorem (Ieong et al., 2005)

For singleton congestion games, better response dynamics (BRD) terminate in at most $n^2 m$ steps (with n #players and m #resources).

- *Proof on next slide. Remember that* $\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k)$.

Lemma

If cost functions (c_e) are integer-valued, then Rosenthal's potential Φ is integer-valued,

Singleton congestion games

Definition

A singleton congestion game $\Gamma = (N, E, (\mathcal{S}_i), (c_e))$ has the property that $\mathcal{S}_i \subseteq \{\{e_1,\}, \{e_2\}, \ldots, \{e_m\}\}$, i.e., every possible strategy consists of a single resource.

Theorem (Ieong et al., 2005)

For singleton congestion games, better response dynamics (BRD) terminate in at most n^2m steps (with n #players and m #resources).

- *Proof on next slide. Remember that $\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k)$.*

Lemma

If cost functions (c_e) are integer-valued, then Rosenthal's potential Φ is integer-valued, and BRD converge in at most $\Phi_{\max} - \Phi_{\min}$ steps.

Singleton congestion games

Definition

A singleton congestion game $\Gamma = (N, E, (\mathcal{S}_i), (c_e))$ has the property that $\mathcal{S}_i \subseteq \{\{e_1,\}, \{e_2\}, \ldots, \{e_m\}\}$, i.e., every possible strategy consists of a single resource.

Theorem (Ieong et al., 2005)

For singleton congestion games, better response dynamics (BRD) terminate in at most $n^2 m$ steps (with n #players and m #resources).

- *Proof on next slide. Remember that $\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k)$.*

Lemma

If cost functions (c_e) are integer-valued, then Rosenthal's potential Φ is integer-valued, and BRD converge in at most $\Phi_{\max} - \Phi_{\min}$ steps.

- $\Phi_{\max}, \Phi_{\min}$ are max. and min. attained by Φ, respectively.
  - For any strategy profile s, it holds that $\Phi_{\min} \leq \Phi(s) \leq \Phi_{\max}$.

Proof idea: Show that cost functions can be replaced by 'nice' (polynomially bounded, integer) cost functions while preserving improving moves.

Proof idea: Show that cost functions can be replaced by 'nice' (polynomially bounded, integer) cost functions while preserving improving moves. *Then apply lemma from previous slide.*

Proof idea: Show that cost functions can be replaced by 'nice' (polynomially bounded, integer) cost functions while preserving improving moves. *Then apply lemma from previous slide.*

Step 1: Defining the 'nice' cost functions.
Consider $C = \bigcup_{e \in E} \{c_e(1), \ldots, c_e(n)\}$.

Proof idea: Show that cost functions can be replaced by 'nice' (polynomially bounded, integer) cost functions while preserving improving moves. *Then apply lemma from previous slide.*

Step 1: Defining the 'nice' cost functions.

Consider $C = \bigcup_{e \in E} \{c_e(1), \ldots, c_e(n)\}$. Note that $|C| \leq nm$.

Proof idea: Show that cost functions can be replaced by 'nice' (polynomially bounded, integer) cost functions while preserving improving moves. *Then apply lemma from previous slide.*

Step 1: Defining the 'nice' cost functions.

Consider $C = \bigcup_{e \in E} \{c_e(1), \ldots, c_e(n)\}$. Note that $|C| \leq nm$.

- Costs of resources for given loads $x_e \in \{1, \ldots, n\}$.

Proof idea: Show that cost functions can be replaced by 'nice' (polynomially bounded, integer) cost functions while preserving improving moves. *Then apply lemma from previous slide.*

Step 1: Defining the 'nice' cost functions.

Consider $C = \bigcup_{e \in E}\{c_e(1), \ldots, c_e(n)\}$. Note that $|C| \leq nm$.

- Costs of resources for given loads $x_e \in \{1, \ldots, n\}$.

For $e \in E$, define $\tilde{c}_e : \{1, \ldots, n\} \to \{1, \ldots, nm\}$ by

$$\tilde{c}_e(i) = r \Leftrightarrow \; r - 1 \text{ distinct values } c_f(j) \in C \text{ for which } c_f(j) < c_e(i).$$

Proof idea: Show that cost functions can be replaced by 'nice' (polynomially bounded, integer) cost functions while preserving improving moves. *Then apply lemma from previous slide.*

Step 1: Defining the 'nice' cost functions.
Consider $C = \bigcup_{e \in E} \{c_e(1), \ldots, c_e(n)\}$. Note that $|C| \leq nm$.

- Costs of resources for given loads $x_e \in \{1, \ldots, n\}$.

For $e \in E$, define $\tilde{c}_e : \{1, \ldots, n\} \to \{1, \ldots, nm\}$ by

$$\tilde{c}_e(i) = r \Leftrightarrow \ r - 1 \text{ distinct values } c_f(j) \in C \text{ for which } c_f(j) < c_e(i).$$

- That is, $c_e(i)$ is the r-th largest number in C.

Proof idea: Show that cost functions can be replaced by 'nice' (polynomially bounded, integer) cost functions while preserving improving moves. *Then apply lemma from previous slide.*

Step 1: Defining the 'nice' cost functions.

Consider $C = \bigcup_{e \in E}\{c_e(1), \ldots, c_e(n)\}$. Note that $|C| \leq nm$.

- Costs of resources for given loads $x_e \in \{1, \ldots, n\}$.

For $e \in E$, define $\tilde{c}_e : \{1, \ldots, n\} \to \{1, \ldots, nm\}$ by

$$\tilde{c}_e(i) = r \Leftrightarrow \; r - 1 \text{ distinct values } c_f(j) \in C \text{ for which } c_f(j) < c_e(i).$$

- That is, $c_e(i)$ is the r-th largest number in C.

Example ($n = 3$ and $m = 2$):

Proof idea: Show that cost functions can be replaced by 'nice' (polynomially bounded, integer) cost functions while preserving improving moves. *Then apply lemma from previous slide.*

Step 1: Defining the 'nice' cost functions.

Consider $C = \bigcup_{e \in E} \{c_e(1), \dots, c_e(n)\}$. Note that $|C| \leq nm$.

- Costs of resources for given loads $x_e \in \{1, \dots, n\}$.

For $e \in E$, define $\tilde{c}_e : \{1, \dots, n\} \to \{1, \dots, nm\}$ by

$$\tilde{c}_e(i) = r \Leftrightarrow \; r - 1 \text{ distinct values } c_f(j) \in C \text{ for which } c_f(j) < c_e(i).$$

- That is, $c_e(i)$ is the r-th largest number in C.

Example ($n = 3$ and $m = 2$):

- $c_1(1) = 3, c_1(2) = 10$, $c_1(3) = 1000$, $c_2(1) = 5$, $c_2(2) = 1000$, $c_2(3) = 1004$.

Proof idea: Show that cost functions can be replaced by 'nice' (polynomially bounded, integer) cost functions while preserving improving moves. *Then apply lemma from previous slide.*

Step 1: Defining the 'nice' cost functions.

Consider $C = \bigcup_{e \in E}\{c_e(1), \ldots, c_e(n)\}$. Note that $|C| \leq nm$.

- Costs of resources for given loads $x_e \in \{1, \ldots, n\}$.

For $e \in E$, define $\tilde{c}_e : \{1, \ldots, n\} \to \{1, \ldots, nm\}$ by

$$\tilde{c}_e(i) = r \Leftrightarrow \ r - 1 \text{ distinct values } c_f(j) \in C \text{ for which } c_f(j) < c_e(i).$$

- That is, $c_e(i)$ is the r-th largest number in C.

Example ($n = 3$ and $m = 2$):

- $c_1(1) = 3, c_1(2) = 10$, $c_1(3) = 1000$, $c_2(1) = 5$, $c_2(2) = 1000$, $c_2(3) = 1004$. We have $C = \{3, 5, 10, 1000, 1004\}$.

Proof idea: Show that cost functions can be replaced by 'nice' (polynomially bounded, integer) cost functions while preserving improving moves. *Then apply lemma from previous slide.*

Step 1: Defining the 'nice' cost functions.

Consider $C = \bigcup_{e \in E} \{c_e(1), \dots, c_e(n)\}$. Note that $|C| \leq nm$.

- Costs of resources for given loads $x_e \in \{1, \dots, n\}$.

For $e \in E$, define $\tilde{c}_e : \{1, \dots, n\} \to \{1, \dots, nm\}$ by

$$\tilde{c}_e(i) = r \Leftrightarrow \ r - 1 \text{ distinct values } c_f(j) \in C \text{ for which } c_f(j) < c_e(i).$$

- That is, $c_e(i)$ is the r-th largest number in C.

Example ($n = 3$ and $m = 2$):

- $c_1(1) = 3, c_1(2) = 10$, $c_1(3) = 1000$, $c_2(1) = 5$, $c_2(2) = 1000$, $c_2(3) = 1004$. We have $C = \{3, 5, 10, 1000, 1004\}$.
- Then $\tilde{c}_1(1) = 1$, $\tilde{c}_1(2) = 3$, $\tilde{c}_1(3) = 4$, $\tilde{c}_2(1) = 2$, $\tilde{c}_2(2) = 4$, $\tilde{c}_2(3) = 5$.

Proof idea: Show that cost functions can be replaced by 'nice' (polynomially bounded, integer) cost functions while preserving improving moves. *Then apply lemma from previous slide.*

Step 1: Defining the 'nice' cost functions.

Consider $C = \bigcup_{e \in E} \{c_e(1), \dots, c_e(n)\}$. Note that $|C| \leq nm$.

- Costs of resources for given loads $x_e \in \{1, \dots, n\}$.

For $e \in E$, define $\tilde{c}_e : \{1, \dots, n\} \to \{1, \dots, nm\}$ by

$$\tilde{c}_e(i) = r \Leftrightarrow \ r - 1 \text{ distinct values } c_f(j) \in C \text{ for which } c_f(j) < c_e(i).$$

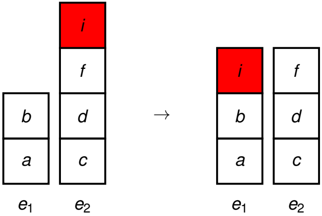
- That is, $c_e(i)$ is the r-th largest number in C.

Example ($n = 3$ and $m = 2$):

- $c_1(1) = 3, c_1(2) = 10$, $c_1(3) = 1000$, $c_2(1) = 5$, $c_2(2) = 1000$, $c_2(3) = 1004$. We have $C = \{3, 5, 10, 1000, 1004\}$.
- Then $\tilde{c}_1(1) = 1$, $\tilde{c}_1(2) = 3$, $\tilde{c}_1(3) = 4$, $\tilde{c}_2(1) = 2$, $\tilde{c}_2(2) = 4$, $\tilde{c}_2(3) = 5$. We have $\tilde{C} = \{1, 2, 3, 4, 5\}$.

Improving moves are preserved under this transformation from c_e to $\tilde{c}_e$.

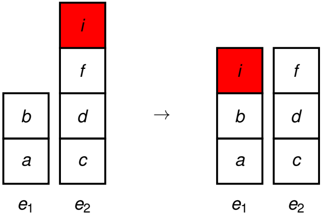
Improving moves are preserved under this transformation from c_e to $\tilde{c}_e$.



In example above, for strategy profile s (on the left),

$$C_i(s'_i, s_{-i}) < C_i(s) \quad \Leftrightarrow \quad \tilde{C}_i(s'_i, s_{-i}) < \tilde{C}_i(s) \tag{1}$$

Improving moves are preserved under this transformation from c_e to $\tilde{c}_e$.



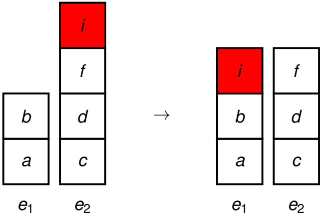
In example above, for strategy profile s (on the left),

$$C_i(s_i', s_{-i}) < C_i(s) \quad \Leftrightarrow \quad \tilde{C}_i(s_i', s_{-i}) < \tilde{C}_i(s) \tag{1}$$

which here means,

$$c_1(x_1(s) + 1) < c_2(x_2(s)) \quad \Leftrightarrow \quad \tilde{c}_1(x_1(s) + 1) < \tilde{c}_2(x_2(s)).$$

Improving moves are preserved under this transformation from c_e to $\tilde{c}_e$.



In example above, for strategy profile s (on the left),

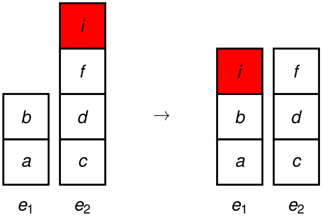
$$C_i(s'_i, s_{-i}) < C_i(s) \quad \Leftrightarrow \quad \tilde{C}_i(s'_i, s_{-i}) < \tilde{C}_i(s) \tag{1}$$

which here means,

$$c_1(x_1(s)+1) < c_2(x_2(s)) \quad \Leftrightarrow \quad \tilde{c}_1(x_1(s)+1) < \tilde{c}_2(x_2(s)).$$

- *Player i has improving move from resource e_2 to e_1 under cost functions (c_e) if and only if it is an improving move under the $(\tilde{c}_e)$.*

Improving moves are preserved under this transformation from c_e to $\tilde{c}_e$.



In example above, for strategy profile s (on the left),

$$C_i(s'_i, s_{-i}) < C_i(s) \quad \Leftrightarrow \quad \tilde{C}_i(s'_i, s_{-i}) < \tilde{C}_i(s) \tag{1}$$

which here means,

$$c_1(x_1(s)+1) < c_2(x_2(s)) \quad \Leftrightarrow \quad \tilde{c}_1(x_1(s)+1) < \tilde{c}_2(x_2(s)).$$

- *Player i has improving move from resource e_2 to e_1 under cost functions (c_e) if and only if it is an improving move under the $(\tilde{c}_e)$.*

Exercise: Show that this transformation fails for non-singleton congestion games (i.e., in general (1) is not true).

Step 2: BRD analysis in 'nice' game.
Rosenthal's potential

$$\tilde{\Phi}(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} \tilde{c}_e(x_e)$$

is integer-valued

Step 2: BRD analysis in 'nice' game.
Rosenthal's potential

$$\tilde{\Phi}(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} \tilde{c}_e(x_e)$$

is integer-valued and satisfies

$$0 \leq \tilde{\Phi} \leq n^2 m.$$

Step 2: BRD analysis in 'nice' game.
Rosenthal's potential

$$\tilde{\Phi}(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} \tilde{c}_e(x_e)$$

is integer-valued and satisfies

$$0 \leq \tilde{\Phi} \leq n^2 m.$$

Why?

Step 2: BRD analysis in 'nice' game.
Rosenthal's potential

$$\tilde{\Phi}(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} \tilde{c}_e(x_e)$$

is integer-valued and satisfies

$$0 \leq \tilde{\Phi} \leq n^2 m.$$

Why?

- First note that $\tilde{c}_e(x_e) \leq nm$ for any load $x_e \in \{1, \ldots, n\}$.

Step 2: BRD analysis in 'nice' game.

Rosenthal's potential

$$\tilde{\Phi}(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} \tilde{c}_e(x_e)$$

is integer-valued and satisfies

$$0 \leq \tilde{\Phi} \leq n^2 m.$$

Why?

- First note that $\tilde{c}_e(x_e) \leq nm$ for any load $x_e \in \{1, \ldots, n\}$.
    - Because $|C| \leq nm$.

Step 2: BRD analysis in 'nice' game.
Rosenthal's potential

$$\tilde{\Phi}(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} \tilde{c}_e(x_e)$$

is integer-valued and satisfies

$$0 \leq \tilde{\Phi} \leq n^2 m.$$

Why?

- First note that $\tilde{c}_e(x_e) \leq nm$ for any load $x_e \in \{1, \dots, n\}$.
    - Because $|C| \leq nm$.
- Also, $\tilde{\Phi}$ is sum of at most n values in $\tilde{C} = \bigcup_{e \in E} \{\tilde{c}_e(1), \dots, \tilde{c}_e(n)\}$.

Step 2: BRD analysis in 'nice' game.

Rosenthal's potential

$$\tilde{\Phi}(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} \tilde{c}_e(x_e)$$

is integer-valued and satisfies

$$0 \leq \tilde{\Phi} \leq n^2 m.$$

Why?

- First note that $\tilde{c}_e(x_e) \leq nm$ for any load $x_e \in \{1, \ldots, n\}$.
  - Because $|C| \leq nm$.
- Also, $\tilde{\Phi}$ is sum of at most n values in $\tilde{C} = \bigcup_{e \in E} \{\tilde{c}_e(1), \ldots, \tilde{c}_e(n)\}$.
  - E.g., $\tilde{\Phi}(s) = [\tilde{c}_1(1) + \tilde{c}_1(2)] + [\tilde{c}_2(1) + \tilde{c}_2(2) + \tilde{c}_2(3) + \tilde{c}_2(4)]$.
    - Sum of $n = 6$ terms.

Step 2: BRD analysis in 'nice' game.
Rosenthal's potential

$$\tilde{\Phi}(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} \tilde{c}_e(x_e)$$

is integer-valued and satisfies

$$0 \le \tilde{\Phi} \le n^2 m.$$

Why?

- First note that $\tilde{c}_e(x_e) \le nm$ for any load $x_e \in \{1, \ldots, n\}$.
  - Because $|C| \le nm$.
- Also, $\tilde{\Phi}$ is sum of at most n values in $\tilde{C} = \bigcup_{e \in E} \{\tilde{c}_e(1), \ldots, \tilde{c}_e(n)\}$.
  - E.g., $\tilde{\Phi}(s) = [\tilde{c}_1(1) + \tilde{c}_1(2)] + [\tilde{c}_2(1) + \tilde{c}_2(2) + \tilde{c}_2(3) + \tilde{c}_2(4)]$.
    - Sum of $n = 6$ terms.
  - That is, in singleton games, we have $\sum_e x_e(s) = n$.

Step 2: BRD analysis in 'nice' game.
Rosenthal's potential

$$\tilde{\Phi}(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} \tilde{c}_e(x_e)$$

is integer-valued and satisfies

$$0 \leq \tilde{\Phi} \leq n^2 m.$$

Why?

- First note that $\tilde{c}_e(x_e) \leq nm$ for any load $x_e \in \{1, \ldots, n\}$.
  - Because $|C| \leq nm$.
- Also, $\tilde{\Phi}$ is sum of at most n values in $\tilde{C} = \bigcup_{e \in E} \{\tilde{c}_e(1), \ldots, \tilde{c}_e(n)\}$.
  - E.g., $\tilde{\Phi}(s) = [\tilde{c}_1(1) + \tilde{c}_1(2)] + [\tilde{c}_2(1) + \tilde{c}_2(2) + \tilde{c}_2(3) + \tilde{c}_2(4)]$.
    - Sum of $n = 6$ terms.
  - That is, in singleton games, we have $\sum_e x_e(s) = n$.

Then apply lemma from Slide 17. $\square$

Symmetric network congestion games

I.e., the “atomic selfish routing game” example from earlier.

Symmetric network congestion games

I.e., the "atomic selfish routing game" example from earlier.

- Resources are edges of given directed graph $G = (V, E)$.
- Common strategy set of players is set of all o, d-paths in G.

Symmetric network congestion games

I.e., the "atomic selfish routing game" example from earlier.

- Resources are edges of given directed graph $G = (V, E)$.
- Common strategy set of players is set of all o, d-paths in G.

Theorem (Best response dynamics)

Best response dynamics might take an exponential (in n) number of steps to terminate (i.e, to converge to a PNE).

Symmetric network congestion games

I.e., the "atomic selfish routing game" example from earlier.

- Resources are edges of given directed graph $G = (V, E)$.
- Common strategy set of players is set of all o, d-paths in G.

Theorem (Best response dynamics)

Best response dynamics might take an exponential (in n) number of steps to terminate (i.e, to converge to a PNE).

Is there another way to compute a PNE?

Symmetric network congestion games

I.e., the "atomic selfish routing game" example from earlier.

- Resources are edges of given directed graph $G = (V, E)$.
- Common strategy set of players is set of all o, d-paths in G.

Theorem (Best response dynamics)

Best response dynamics might take an exponential (in n) number of steps to terminate (i.e, to converge to a PNE).

Is there another way to compute a PNE?

Theorem (Fabrikant et al., 2004)

There exists a poly(n, m)-time algorithm for computing a PNE in a symmetric network congestion game when the cost functions are non-negative and non-decreasing.

Symmetric network congestion games

I.e., the "atomic selfish routing game" example from earlier.

- Resources are edges of given directed graph $G = (V, E)$.
- Common strategy set of players is set of all o, d-paths in G.

Theorem (Best response dynamics)

Best response dynamics might take an exponential (in n) number of steps to terminate (i.e, to converge to a PNE).

Is there another way to compute a PNE?

Theorem (Fabrikant et al., 2004)

There exists a poly(n, m)-time algorithm for computing a PNE in a symmetric network congestion game when the cost functions are non-negative and non-decreasing.

- Idea: Compute strategy profile s minimizing Rosenthal's potential.

Symmetric network congestion games

I.e., the “atomic selfish routing game” example from earlier.

- Resources are edges of given directed graph $G = (V, E)$.
- Common strategy set of players is set of all o, d-paths in G.

Theorem (Best response dynamics)

Best response dynamics might take an exponential (in n) number of steps to terminate (i.e, to converge to a PNE).

Is there another way to compute a PNE?

Theorem (Fabrikant et al., 2004)

There exists a poly(n, m)-time algorithm for computing a PNE in a symmetric network congestion game when the cost functions are non-negative and non-decreasing.

- Idea: Compute strategy profile s minimizing Rosenthal’s potential.
- Convince yourself this is indeed a pure Nash equilibrium.

Reduction to flow problem

If every player chooses o, d-path, resulting in strategy profile s,

Reduction to flow problem

If every player chooses o, d-path, resulting in strategy profile s, we obtain a so-called o, d-flow of size n.

Reduction to flow problem

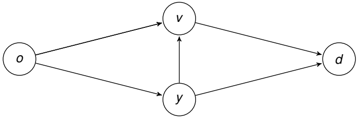
If every player chooses o, d-path, resulting in strategy profile s, we obtain a so-called o, d-flow of size n.

- Every player routes one unit of flow over some path.

Reduction to flow problem

If every player chooses o, d-path, resulting in strategy profile s, we obtain a so-called o, d-flow of size n.

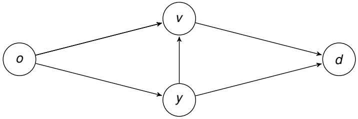
- Every player routes one unit of flow over some path.



Reduction to flow problem

If every player chooses o, d-path, resulting in strategy profile s, we obtain a so-called o, d-flow of size n.

- Every player routes one unit of flow over some path.



Resulting loads $x_e(s) = f_e$ satisfy the linear (in)equalities

$$\mathcal{F} = \Big\{ f \in \mathbb{R}^{|E|}_{\geq 0} : \begin{array}{ll} \displaystyle\sum_{w:(w,v)\in E} f_{wv} = \sum_{w:(v,w)\in E} f_{vw} & \forall v \in V \setminus \{o, d\} \\ \displaystyle\sum_{w:(o,w)\in E} f_{ow} = n & \\ \displaystyle\sum_{w:(w,d)\in E} f_{wd} = n & \\ f_{vw} \geq 0 & \forall (v,w) \in E \Big\}. \end{array}$$

High-level idea: Instead of computing a strategy profile $s^* \in \times_i \mathcal{S}_i$ minimizing

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k),$$

High-level idea: Instead of computing a strategy profile $s^* \in \times_i \mathcal{S}_i$ minimizing

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k),$$

compute an integral o, d-flow (or load profile) $f^* \in \mathcal{F}$ that minimizes

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k).$$

High-level idea: Instead of computing a strategy profile $s^* \in \times_i \mathcal{S}_i$ minimizing

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k),$$

compute an integral o, d-flow (or load profile) $f^* \in \mathcal{F}$ that minimizes

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k).$$

Map o, d-flow f^* to strategy profile s^* minimizing Φ.

High-level idea: Instead of computing a strategy profile $s^* \in \times_i \mathcal{S}_i$ minimizing

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k),$$

compute an integral o, d-flow (or load profile) $f^* \in \mathcal{F}$ that minimizes

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k).$$

Map o, d-flow f^* to strategy profile s^* minimizing Φ. Can we always do this?

High-level idea: Instead of computing a strategy profile $s^* \in \times_i \mathcal{S}_i$ minimizing

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k),$$

compute an integral o, d-flow (or load profile) $f^* \in \mathcal{F}$ that minimizes

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k).$$

Map o, d-flow f^* to strategy profile s^* minimizing Φ. Can we always do this?

Lemma

*Every **integral** $f \in \mathcal{F}$ can be decomposed into n (one for each player) o, d-paths that each contain one unit of flow.*

High-level idea: Instead of computing a strategy profile $s^* \in \times_i \mathcal{S}_i$ minimizing

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k),$$

compute an integral o, d-flow (or load profile) $f^* \in \mathcal{F}$ that minimizes

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k).$$

Map o, d-flow f^* to strategy profile s^* minimizing Φ. Can we always do this?

Lemma

*Every **integral** $f \in \mathcal{F}$ can be decomposed into n (one for each player) o, d-paths that each contain one unit of flow.*

- *(For simplicity, we assume here that $G = (V, E)$ is acyclic.)*

High-level idea: Instead of computing a strategy profile $s^* \in \times_i \mathcal{S}_i$ minimizing

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k),$$

compute an integral o, d-flow (or load profile) $f^* \in \mathcal{F}$ that minimizes

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k).$$

Map o, d-flow f^* to strategy profile s^* minimizing Φ. Can we always do this?

Lemma

*Every **integral** $f \in \mathcal{F}$ can be decomposed into n (one for each player) o, d-paths that each contain one unit of flow.*

- *(For simplicity, we assume here that $G = (V, E)$ is acyclic.)*

Assign resulting paths to players.

High-level idea: Instead of computing a strategy profile $s^* \in \times_i \mathcal{S}_i$ minimizing

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k),$$

compute an integral o, d-flow (or load profile) $f^* \in \mathcal{F}$ that minimizes

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k).$$

Map o, d-flow f^* to strategy profile s^* minimizing Φ. Can we always do this?

Lemma

*Every **integral** $f \in \mathcal{F}$ can be decomposed into n (one for each player) o, d-paths that each contain one unit of flow.*

- *(For simplicity, we assume here that $G = (V, E)$ is acyclic.)*

Assign resulting paths to players. This gives the desired profile s^*.

High-level idea: Instead of computing a strategy profile $s^* \in \times_i \mathcal{S}_i$ minimizing

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k),$$

compute an integral o, d-flow (or load profile) $f^* \in \mathcal{F}$ that minimizes

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k).$$

Map o, d-flow f^* to strategy profile s^* minimizing Φ. Can we always do this?

Lemma

*Every **integral** $f \in \mathcal{F}$ can be decomposed into n (one for each player) o, d-paths that each contain one unit of flow.*

- *(For simplicity, we assume here that $G = (V, E)$ is acyclic.)*

Assign resulting paths to players. This gives the desired profile s^*.

- Does not matter which path is assigned to which player.

High-level idea: Instead of computing a strategy profile $s^* \in \times_i \mathcal{S}_i$ minimizing

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k),$$

compute an integral o, d-flow (or load profile) $f^* \in \mathcal{F}$ that minimizes

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k).$$

Map o, d-flow f^* to strategy profile s^* minimizing Φ. Can we always do this?

Lemma

*Every **integral** $f \in \mathcal{F}$ can be decomposed into n (one for each player) o, d-paths that each contain one unit of flow.*

- *(For simplicity, we assume here that $G = (V, E)$ is acyclic.)*

Assign resulting paths to players. This gives the desired profile s^*.

- Does not matter which path is assigned to which player.
- *Symmetry assumption is crucial here!*

High-level idea: Instead of computing a strategy profile $s^* \in \times_i \mathcal{S}_i$ minimizing

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k),$$

compute an integral o, d-flow (or load profile) $f^* \in \mathcal{F}$ that minimizes

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k).$$

Map o, d-flow f^* to strategy profile s^* minimizing Φ. Can we always do this?

Lemma

*Every **integral** $f \in \mathcal{F}$ can be decomposed into n (one for each player) o, d-paths that each contain one unit of flow.*

- *(For simplicity, we assume here that $G = (V, E)$ is acyclic.)*

Assign resulting paths to players. This gives the desired profile s^*.

- Does not matter which path is assigned to which player.
- *Symmetry assumption is crucial here! (Think about it.)*

High-level idea: Instead of computing a strategy profile $s^* \in \times_i \mathcal{S}_i$ minimizing

$$\Phi(s) = \sum_{e \in E} \sum_{k=1}^{x_e(s)} c_e(k),$$

compute an integral o, d-flow (or load profile) $f^* \in \mathcal{F}$ that minimizes

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k).$$

Map o, d-flow f^* to strategy profile s^* minimizing Φ. Can we always do this?

Lemma

Every ***integral*** *$f \in \mathcal{F}$ can be decomposed into n (one for each player) o, d-paths that each contain one unit of flow.*

- *(For simplicity, we assume here that $G = (V, E)$ is acyclic.)*

Assign resulting paths to players. This gives the desired profile s^*.

- Does not matter which path is assigned to which player.
- *Symmetry assumption is crucial here! (Think about it.)*

Computing profile s^* minimizing Rosenthal's potential Φ:

- Compute integral $f^* \in \mathcal{F}$ that minimizes

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k).$$

- Decompose f^* into n paths, and assign those to players.
  - This gives desired strategy profile s (with $x_e(s) = f_e^* \;\; \forall e \in E$)

Computing profile s^* minimizing Rosenthal's potential Φ:

- Compute integral $f^* \in \mathcal{F}$ that minimizes

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k).$$

- Decompose f^* into n paths, and assign those to players.
  - This gives desired strategy profile s (with $x_e(s) = f_e^* \;\; \forall e \in E$)

How to compute minimizer of $\bar{\Phi}$?

Computing profile s^* minimizing Rosenthal's potential Φ:

- Compute integral $f^* \in \mathcal{F}$ that minimizes

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k).$$

- Decompose f^* into n paths, and assign those to players.
    - This gives desired strategy profile s (with $x_e(s) = f_e^* \;\; \forall e \in E$)

How to compute minimizer of $\bar{\Phi}$?

- Reduction to min-cost flow problem.

Computing profile s^* minimizing Rosenthal's potential Φ:

- Compute integral $f^* \in \mathcal{F}$ that minimizes

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k).$$

- Decompose f^* into n paths, and assign those to players.
  - This gives desired strategy profile s (with $x_e(s) = f_e^* \;\; \forall e \in E$)

How to compute minimizer of $\bar{\Phi}$?

- Reduction to min-cost flow problem.
- Can be solved in poly(n, m) time.

Computing profile s^* minimizing Rosenthal's potential Φ:

- Compute integral $f^* \in \mathcal{F}$ that minimizes

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k).$$

- Decompose f^* into n paths, and assign those to players.
  - This gives desired strategy profile s (with $x_e(s) = f_e^* \;\; \forall e \in E$)

How to compute minimizer of $\bar{\Phi}$?

- Reduction to min-cost flow problem.
- Can be solved in poly(n, m) time.

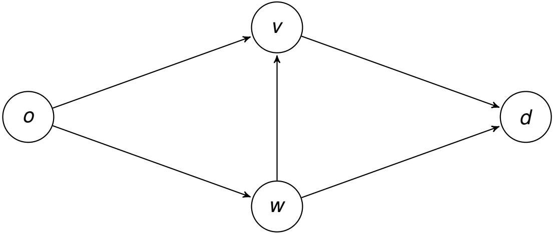
Remark

This high-level approach also works for other congestion games with some 'combinatorial' structure, e.g., (Del Pia-Michini-Ferris, 2015).

Minimum cost flow problem

Directed graph $G = (V, A)$ with origin o and destination d; flow size $n \in \mathbb{Q}$.

- Edge $e = (v, w) \in E$ has capacity u_{vw} and cost k_{vw}.

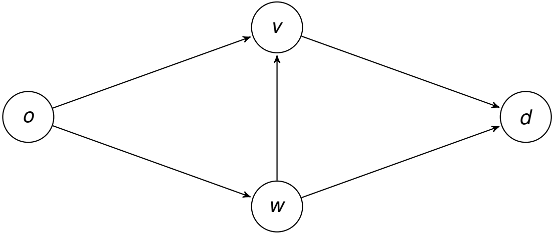


$$
\begin{array}{lll}
\min & \displaystyle\sum_{e=(v,w)\in E} k_{vw} f_{vw} & \\
\text{subject to} & f \in \mathcal{F} & \\
 & f_{vw} \leq u_{vw} & \forall (u, v) \in E
\end{array}
$$

Minimum cost flow problem

Directed graph $G = (V, A)$ with origin o and destination d; flow size $n \in \mathbb{Q}$.

- Edge $e = (v, w) \in E$ has capacity u_{vw} and cost k_{vw}.



$$
\begin{array}{lll}
\min & \displaystyle\sum_{e=(v,w)\in E} k_{vw} f_{vw} & \\
\text{subject to} & f \in \mathcal{F} & \\
 & f_{vw} \leq u_{vw} & \forall (u, v) \in E
\end{array}
$$

Integral flow can be found in poly-time, when capacities are integral.

Reduction to min-cost flow problem (try yourself!)

Problem is that

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k)$$

is not linear in the variables f_e.

Edge-doubling trick ($n = 5$):

Reduction to min-cost flow problem (try yourself!)

Problem is that

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k)$$

is not linear in the variables f_e.

Edge-doubling trick ($n = 5$):

- Introduce copies with capacity 1 and cost $c_e(1), c_e(2), \ldots, c_e(n)$.

Reduction to min-cost flow problem (try yourself!)

Problem is that

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k)$$

is not linear in the variables f_e.

Edge-doubling trick ($n = 5$):

- Introduce copies with capacity 1 and cost $c_e(1), c_e(2), \ldots, c_e(n)$.
    - Remember costs are non-decreasing and non-negative.

Reduction to min-cost flow problem (try yourself!)

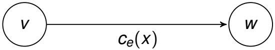
Problem is that

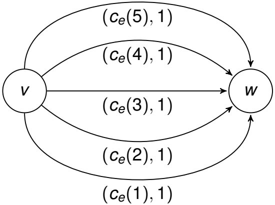
$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k)$$

is not linear in the variables f_e.

Edge-doubling trick ($n = 5$):

- Introduce copies with capacity 1 and cost $c_e(1), c_e(2), \ldots, c_e(n)$.
    - Remember costs are non-decreasing and non-negative.





Reduction to min-cost flow problem (try yourself!)

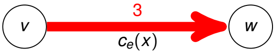
Problem is that

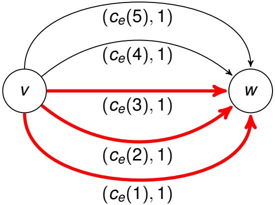
$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k)$$

is not linear in the variables f_e.

Edge-doubling trick ($n = 5$):

- Introduce copies with capacity 1 and cost $c_e(1), c_e(2), \ldots, c_e(n)$.
  - Remember costs are non-decreasing and non-negative.





Reduction to min-cost flow problem (try yourself!)

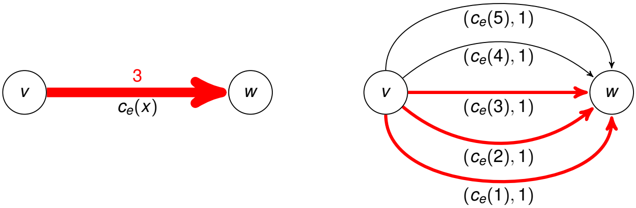
Problem is that

$$\bar{\Phi}(f) = \sum_{e \in E} \sum_{k=1}^{f_e} c_e(k)$$

is not linear in the variables f_e.

Edge-doubling trick ($n = 5$):

- Introduce copies with capacity 1 and cost $c_e(1), c_e(2), \ldots, c_e(n)$.
  - Remember costs are non-decreasing and non-negative.



Every integral min-cost flow of size n in graph with copied edges corresponds to flow minimizing $\bar{\Phi}$.

Local search

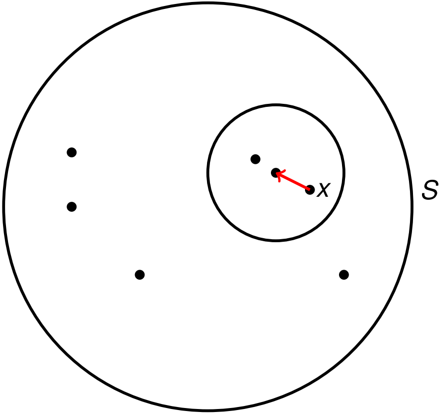
High-level idea

Given function $f : S \to \mathbb{R}$, where S is a finite set.

High-level idea

Given function $f : S \to \mathbb{R}$, where S is a finite set.

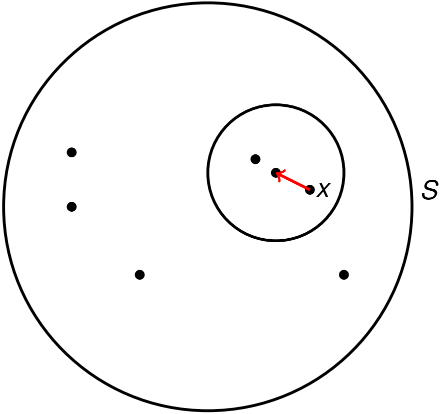
- Can we find 'local' improvement in objective value $f(x)$?



High-level idea

Given function $f : S \to \mathbb{R}$, where S is a finite set.

- Can we find 'local' improvement in objective value $f(x)$?

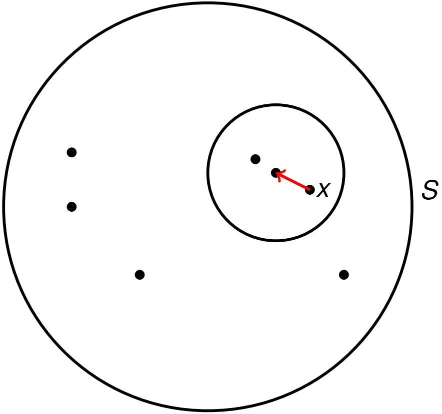


- Recall better response dynamics.

High-level idea

Given function $f : S \to \mathbb{R}$, where S is a finite set.

- Can we find 'local' improvement in objective value $f(x)$?



- Recall better response dynamics.
  - Essentially tries to find local improvement for Rosenthal's potential.

Local search problems

Definition

A **local search problem** Π is given by:

- Set of instances $\mathcal{I}$;

Local search problems

Definition

A **local search problem** Π is given by:

- Set of instances $\mathcal{I}$;
- For every instance $I \in \mathcal{I}$:

Local search problems

Definition

A **local search problem** Π is given by:

- Set of instances $\mathcal{I}$;
- For every instance $I \in \mathcal{I}$:
  - Set $F(I)$ of feasible solutions;

Local search problems

Definition

A **local search problem** Π is given by:

- Set of instances $\mathcal{I}$;
- For every instance $I \in \mathcal{I}$:
  - Set $F(I)$ of feasible solutions;
  - An objective function $\Phi : F(I) \to \mathbb{Z}$;

Local search problems

Definition

A **local search problem** Π is given by:

- Set of instances $\mathcal{I}$;
- For every instance $I \in \mathcal{I}$:
  - Set $F(I)$ of feasible solutions;
  - An objective function $\Phi : F(I) \to \mathbb{Z}$;
  - For every $S \in F(I)$, a neighborhood $\mathcal{N}(S, I) \subseteq F(I)$ of S.

Local search problems

Definition

A **local search problem** Π is given by:

- Set of instances $\mathcal{I}$;
- For every instance $I \in \mathcal{I}$:
  - Set $F(I)$ of feasible solutions;
  - An objective function $\Phi : F(I) \to \mathbb{Z}$;
  - For every $S \in F(I)$, a neighborhood $\mathcal{N}(S, I) \subseteq F(I)$ of S.

Local search problems

Definition

A **local search problem** Π is given by:

- Set of instances $\mathcal{I}$;
- For every instance $I \in \mathcal{I}$:
  - Set $F(I)$ of feasible solutions;
  - An objective function $\Phi : F(I) \to \mathbb{Z}$;
  - For every $S \in F(I)$, a neighborhood $\mathcal{N}(S, I) \subseteq F(I)$ of S.

Goal: Find a feasible solution $S \in F(I)$ that is a **local minimum**:

$$\Phi(S) \leq \Phi(S'), \qquad \forall S' \in \mathcal{N}(S, I).$$

Local search problems

Definition

A **local search problem** Π is given by:

- Set of instances $\mathcal{I}$;
- For every instance $I \in \mathcal{I}$:
  - Set $F(I)$ of feasible solutions;
  - An objective function $\Phi : F(I) \to \mathbb{Z}$;
  - For every $S \in F(I)$, a neighborhood $\mathcal{N}(S, I) \subseteq F(I)$ of S.

Goal: Find a feasible solution $S \in F(I)$ that is a **local minimum**:

$$\Phi(S) \leq \Phi(S'), \qquad \forall S' \in \mathcal{N}(S, I).$$

We are interested in "unilateral deviations" as neighborhood, and Rosenthal's potential as objective function.

Local search problems

Definition

A **local search problem** Π is given by:

- Set of instances $\mathcal{I}$;
- For every instance $I \in \mathcal{I}$:
  - Set $F(I)$ of feasible solutions;
  - An objective function $\Phi : F(I) \to \mathbb{Z}$;
  - For every $S \in F(I)$, a neighborhood $\mathcal{N}(S, I) \subseteq F(I)$ of S.

Goal: Find a feasible solution $S \in F(I)$ that is a **local minimum**:

$$\Phi(S) \leq \Phi(S'), \qquad \forall S' \in \mathcal{N}(S, I).$$

We are interested in "unilateral deviations" as neighborhood, and Rosenthal's potential as objective function. PNEs are then precisely the local minima.

Complexity Class PLS

Definition

A local search problem Π belongs to the complexity class PLS (polynomial local search) if for every instance $I \in \mathcal{I}$ the following can be done in polynomial time:

Complexity Class PLS

Definition

A local search problem Π belongs to the complexity class PLS (polynomial local search) if for every instance $I \in \mathcal{I}$ the following can be done in polynomial time:

- Compute an initial feasible solution $S \in F(I)$;

Complexity Class PLS

Definition

A local search problem Π belongs to the complexity class PLS (polynomial local search) if for every instance $I \in \mathcal{I}$ the following can be done in polynomial time:

- Compute an initial feasible solution $S \in F(I)$;
- For a given solution $S \in F(I)$:

Complexity Class PLS

Definition

A local search problem Π belongs to the complexity class PLS (polynomial local search) if for every instance $I \in \mathcal{I}$ the following can be done in polynomial time:

- Compute an initial feasible solution $S \in F(I)$;
- For a given solution $S \in F(I)$:
  - Compute $\Phi(S)$;

Complexity Class PLS

Definition

A local search problem Π belongs to the complexity class PLS (polynomial local search) if for every instance $I \in \mathcal{I}$ the following can be done in polynomial time:

- Compute an initial feasible solution $S \in F(I)$;
- For a given solution $S \in F(I)$:
  - Compute $\Phi(S)$;
  - Determine whether S is a local minimum;

Complexity Class PLS

Definition

A local search problem Π belongs to the complexity class PLS (polynomial local search) if for every instance $I \in \mathcal{I}$ the following can be done in polynomial time:

- Compute an initial feasible solution $S \in F(I)$;
- For a given solution $S \in F(I)$:
  - Compute $\Phi(S)$;
  - Determine whether S is a local minimum;
  - If S is not a local minimum, find a better solution S' in the neighborhood of S,

Complexity Class PLS

Definition

A local search problem Π belongs to the complexity class PLS (polynomial local search) if for every instance $I \in \mathcal{I}$ the following can be done in polynomial time:

- Compute an initial feasible solution $S \in F(I)$;
- For a given solution $S \in F(I)$:
  - Compute $\Phi(S)$;
  - Determine whether S is a local minimum;
  - If S is not a local minimum, find a better solution S' in the neighborhood of S, i.e., $S' \in \mathcal{N}(S, I)$ with $\Phi(S') < \Phi(S)$.

Complexity Class PLS

Definition

A local search problem Π belongs to the complexity class PLS (polynomial local search) if for every instance $I \in \mathcal{I}$ the following can be done in polynomial time:

- Compute an initial feasible solution $S \in F(I)$;
- For a given solution $S \in F(I)$:
  - Compute $\Phi(S)$;
  - Determine whether S is a local minimum;
  - If S is not a local minimum, find a better solution S' in the neighborhood of S, i.e., $S' \in \mathcal{N}(S, I)$ with $\Phi(S') < \Phi(S)$.

Complexity Class PLS

Definition

A local search problem Π belongs to the complexity class PLS (polynomial local search) if for every instance $I \in \mathcal{I}$ the following can be done in polynomial time:

- Compute an initial feasible solution $S \in F(I)$;
- For a given solution $S \in F(I)$:
  - Compute $\Phi(S)$;
  - Determine whether S is a local minimum;
  - If S is not a local minimum, find a better solution S' in the neighborhood of S, i.e., $S' \in \mathcal{N}(S, I)$ with $\Phi(S') < \Phi(S)$.

The procedure in which one repeatedly tries to find a better solution in the neighborhood is known as local search.

Maximum cut

Max-cut

Given undirected graph $G = (V, E)$ and weight function $w : E \to \mathbb{R}_{\geq 0}$, find partition $V = S \cup \bar{S}$ that maximizes

$$\alpha(S, \bar{S}) = \sum_{e=\{i,j\}:i \in S, j \in \bar{S}} w_{ij}.$$

Maximum cut

Max-cut

Given undirected graph $G = (V, E)$ and weight function $w : E \to \mathbb{R}_{\geq 0}$, find partition $V = S \cup \bar{S}$ that maximizes

$$\alpha(S, \bar{S}) = \sum_{e=\{i,j\}:i\in S, j\in \bar{S}} w_{ij}.$$

Local Search: FLIP neighborhood

For cut $(S, \bar{S})$ its neighbourhood is given by all $(T, \bar{T})$ that can be obtained by flipping precisely one node to its other side in $(S, \bar{S})$.

Maximum cut

Max-cut

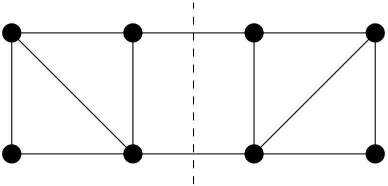
Given undirected graph $G = (V, E)$ and weight function $w : E \to \mathbb{R}_{\geq 0}$, find partition $V = S \cup \bar{S}$ that maximizes

$$\alpha(S, \bar{S}) = \sum_{e=\{i,j\}: i \in S, j \in \bar{S}} w_{ij}.$$

Local Search: FLIP neighborhood

For cut $(S, \bar{S})$ its neighbourhood is given by all $(T, \bar{T})$ that can be obtained by flipping precisely one node to its other side in $(S, \bar{S})$.

S



$\bar{S}$

$\alpha(S, \bar{S}) = 2$

Maximum cut

Max-cut

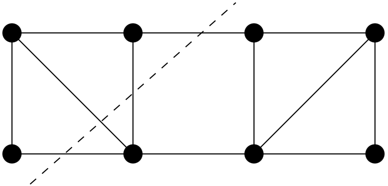
Given undirected graph $G = (V, E)$ and weight function $w : E \to \mathbb{R}_{\geq 0}$, find partition $V = S \cup \bar{S}$ that maximizes

$$\alpha(S, \bar{S}) = \sum_{e=\{i,j\}: i \in S, j \in \bar{S}} w_{ij}.$$

Local Search: FLIP neighborhood

For cut $(S, \bar{S})$ its neighbourhood is given by all $(T, \bar{T})$ that can be obtained by flipping precisely one node to its other side in $(S, \bar{S})$.

T



$\bar{T}$

$\alpha(T, \bar{T}) = 4$

PLS-reduction

PLS-reduction

"Problem Π_1 can be reduced to Π_2" means that Π_1 can be modeled as a special case of Π_2,

PLS-reduction

"Problem Π_1 can be reduced to Π_2" means that Π_1 can be modeled as a special case of Π_2, Hence, Π_2 is the "more difficult" problem of the two

PLS-reduction

"Problem Π_1 *can be reduced to* Π_2*" means that* Π_1 *can be modeled as a special case of* Π_2*, Hence,* Π_2 *is the "more difficult" problem of the two (i.e., not easier than the other).*

PLS-reduction

"Problem Π_1 can be reduced to Π_2" means that Π_1 can be modeled as a special case of Π_2, Hence, Π_2 is the "more difficult" problem of the two (i.e., not easier than the other).

Definition

Let $\Pi_1 = (\mathcal{I}_1, F_1, \Phi_1, \mathcal{N}_1)$ and $\Pi_2 = (\mathcal{I}_2, F_2, \Phi_2, \mathcal{N}_2)$ be two local search problems in PLS.

PLS-reduction

“Problem Π_1 can be reduced to Π_2” means that Π_1 can be modeled as a special case of Π_2, Hence, Π_2 is the "more difficult" problem of the two (i.e., not easier than the other).

Definition

Let $\Pi_1 = (\mathcal{I}_1, F_1, \Phi_1, \mathcal{N}_1)$ and $\Pi_2 = (\mathcal{I}_2, F_2, \Phi_2, \mathcal{N}_2)$ be two local search problems in PLS. Π_1 is PLS-reducible to Π_2 if there are two polynomial time computable functions f and g such that

PLS-reduction

"Problem Π_1 can be reduced to Π_2" means that Π_1 can be modeled as a special case of Π_2, Hence, Π_2 is the "more difficult" problem of the two (i.e., not easier than the other).

Definition

Let $\Pi_1 = (\mathcal{I}_1, F_1, \Phi_1, \mathcal{N}_1)$ and $\Pi_2 = (\mathcal{I}_2, F_2, \Phi_2, \mathcal{N}_2)$ be two local search problems in PLS. Π_1 is PLS-reducible to Π_2 if there are two polynomial time computable functions f and g such that

- f maps every instance $I \in \mathcal{I}_1$ of Π_1 to an instance $f(I) \in \mathcal{I}_2$ of Π_2;

PLS-reduction

"Problem Π_1 can be reduced to Π_2" means that Π_1 can be modeled as a special case of Π_2, Hence, Π_2 is the "more difficult" problem of the two (i.e., not easier than the other).

Definition

Let $\Pi_1 = (\mathcal{I}_1, F_1, \Phi_1, \mathcal{N}_1)$ and $\Pi_2 = (\mathcal{I}_2, F_2, \Phi_2, \mathcal{N}_2)$ be two local search problems in PLS. Π_1 is PLS-reducible to Π_2 if there are two polynomial time computable functions f and g such that

- f maps every instance $I \in \mathcal{I}_1$ of Π_1 to an instance $f(I) \in \mathcal{I}_2$ of Π_2;
- g maps every tuple (S_2, I) with $S_2 \in F_2(f(I))$ to a solution $S_1 \in F_1(I)$;

PLS-reduction

"Problem Π_1 can be reduced to Π_2" means that Π_1 can be modeled as a special case of Π_2, Hence, Π_2 is the "more difficult" problem of the two (i.e., not easier than the other).

Definition

Let $\Pi_1 = (\mathcal{I}_1, F_1, \Phi_1, \mathcal{N}_1)$ and $\Pi_2 = (\mathcal{I}_2, F_2, \Phi_2, \mathcal{N}_2)$ be two local search problems in PLS. Π_1 is PLS-reducible to Π_2 if there are two polynomial time computable functions f and g such that

- f maps every instance $I \in \mathcal{I}_1$ of Π_1 to an instance $f(I) \in \mathcal{I}_2$ of Π_2;
- g maps every tuple (S_2, I) with $S_2 \in F_2(f(I))$ to a solution $S_1 \in F_1(I)$; (**Feasible solutions map to feasible solutions.**)

PLS-reduction

"Problem Π_1 can be reduced to Π_2" means that Π_1 can be modeled as a special case of Π_2, Hence, Π_2 is the "more difficult" problem of the two (i.e., not easier than the other).

Definition

Let $\Pi_1 = (\mathcal{I}_1, F_1, \Phi_1, \mathcal{N}_1)$ and $\Pi_2 = (\mathcal{I}_2, F_2, \Phi_2, \mathcal{N}_2)$ be two local search problems in PLS. Π_1 is PLS-reducible to Π_2 if there are two polynomial time computable functions f and g such that

- f maps every instance $I \in \mathcal{I}_1$ of Π_1 to an instance $f(I) \in \mathcal{I}_2$ of Π_2;
- g maps every tuple (S_2, I) with $S_2 \in F_2(f(I))$ to a solution $S_1 \in F_1(I)$; (**Feasible solutions map to feasible solutions.**)
- for all $I \in \mathcal{I}_1$: if S_2 is a local minimum of $f(I)$, then $g(S_2, I)$ is a local minimum of I.

PLS-reduction

"Problem Π_1 can be reduced to Π_2" means that Π_1 can be modeled as a special case of Π_2, Hence, Π_2 is the "more difficult" problem of the two (i.e., not easier than the other).

Definition

Let $\Pi_1 = (\mathcal{I}_1, F_1, \Phi_1, \mathcal{N}_1)$ and $\Pi_2 = (\mathcal{I}_2, F_2, \Phi_2, \mathcal{N}_2)$ be two local search problems in PLS. Π_1 is PLS-reducible to Π_2 if there are two polynomial time computable functions f and g such that

- f maps every instance $I \in \mathcal{I}_1$ of Π_1 to an instance $f(I) \in \mathcal{I}_2$ of Π_2;
- g maps every tuple (S_2, I) with $S_2 \in F_2(f(I))$ to a solution $S_1 \in F_1(I)$; (**Feasible solutions map to feasible solutions.**)
- for all $I \in \mathcal{I}_1$: if S_2 is a local minimum of $f(I)$, then $g(S_2, I)$ is a local minimum of I. (**Local minima map to local minima.**)

PLS-completeness

Definition

A local search problem Π is PLS-complete if

- Π belongs to the complexity class PLS;
- every problem in PLS is PLS-reducible to Π.

PLS-completeness

Definition

A local search problem Π is PLS-complete if

- Π belongs to the complexity class PLS;
- every problem in PLS is PLS-reducible to Π.

Implication: If there is a polynomial time algorithm that computes a local optimum for a PLS-complete problem Π,

PLS-completeness

Definition

A local search problem Π is PLS-complete if

- Π belongs to the complexity class PLS;
- every problem in PLS is PLS-reducible to Π.

Implication: If there is a polynomial time algorithm that computes a local optimum for a PLS-complete problem Π, then there exists a polynomial time algorithm for finding a local optimum for every problem in PLS.

PLS-completeness

Definition

A local search problem Π is PLS-complete if

- Π belongs to the complexity class PLS;
- every problem in PLS is PLS-reducible to Π.

Implication: If there is a polynomial time algorithm that computes a local optimum for a PLS-complete problem Π, then there exists a polynomial time algorithm for finding a local optimum for every problem in PLS.

Remark

The definition of PLS does not require you to solve a PLS(-complete) problem with local search.

From max-cut to PNE in congestion games

Theorem

Maximum cut with FLIP neighborhood is PLS-complete.

From max-cut to PNE in congestion games

Theorem

Maximum cut with FLIP neighborhood is PLS-complete.

- In particular, local search might take an exponential long time to converge to a local optimum.

From max-cut to PNE in congestion games

Theorem

Maximum cut with FLIP neighborhood is PLS-complete.

- In particular, local search might take an exponential long time to converge to a local optimum.

Theorem

Computing PNE with "unilateral deviation" neighborhood and, Rosenthal's potential as objective function, is PLS-complete.

From max-cut to PNE in congestion games

Theorem

Maximum cut with FLIP neighborhood is PLS-complete.

- In particular, local search might take an exponential long time to converge to a local optimum.

Theorem

Computing PNE with "unilateral deviation" neighborhood and, Rosenthal's potential as objective function, is PLS-complete.

- Unilateral deviation neighborhood of $s \in \times_i \mathcal{S}_i$ is given by

$$\mathcal{N}(s) = \bigcup_i \{(s_i', s_{-i}) : s_i' \in \mathcal{S}_i\}$$

i.e., all profiles that can be obtained by letting at most one player deviate to another strategy.

From max-cut to PNE in congestion games

Theorem

Maximum cut with FLIP neighborhood is PLS-complete.

- In particular, local search might take an exponential long time to converge to a local optimum.

Theorem

Computing PNE with "unilateral deviation" neighborhood and, Rosenthal's potential as objective function, is PLS-complete.

- Unilateral deviation neighborhood of $s \in \times_i \mathcal{S}_i$ is given by

$$\mathcal{N}(s) = \bigcup_i \{(s'_i, s_{-i}) : s'_i \in \mathcal{S}_i\}$$

  i.e., all profiles that can be obtained by letting at most one player deviate to another strategy.
- Reduction from Max-cut with FLIP neighborhoods.

Sketch of reduction

Let $\mathcal{I} = (G, w)$ be an instance of max-cut with FLIP neighborhood on graph $G = (V, E)$, with edge-weight function w.

Sketch of reduction

Let $\mathcal{I} = (G, w)$ be an instance of max-cut with FLIP neighborhood on graph $G = (V, E)$, with edge-weight function w.

Maximizing weight of cut edges is equivalent to minimizing weight of non-cut edges (also locally).

Sketch of reduction

Let $\mathcal{I} = (G, w)$ be an instance of max-cut with FLIP neighborhood on graph $G = (V, E)$, with edge-weight function w.

Maximizing weight of cut edges is equivalent to minimizing weight of non-cut edges (also locally).

Minimum uncut

Given undirected graph $G = (V, E)$ and weight function $w : E \to \mathbb{R}_{\geq 0}$, find partition $V = S \cup \bar{S}$ that minimizes $\sum_{\{i,j\} \in E : i,j \in S \text{ or } i,j \in \bar{S}} w_e$.

Sketch of reduction

Let $\mathcal{I} = (G, w)$ be an instance of max-cut with FLIP neighborhood on graph $G = (V, E)$, with edge-weight function w.

Maximizing weight of cut edges is equivalent to minimizing weight of non-cut edges (also locally).

Minimum uncut

Given undirected graph $G = (V, E)$ and weight function $w : E \to \mathbb{R}_{\geq 0}$, find partition $V = S \cup \bar{S}$ that minimizes $\sum_{\{i,j\} \in E : i,j \in S \text{ or } i,j \in \bar{S}} w_e$.

Why?

Sketch of reduction

Let $\mathcal{I} = (G, w)$ be an instance of max-cut with FLIP neighborhood on graph $G = (V, E)$, with edge-weight function w.

Maximizing weight of cut edges is equivalent to minimizing weight of non-cut edges (also locally).

Minimum uncut

Given undirected graph $G = (V, E)$ and weight function $w : E \to \mathbb{R}_{\geq 0}$, find partition $V = S \cup \bar{S}$ that minimizes $\sum_{\{i,j\} \in E : i,j \in S \text{ or } i,j \in \bar{S}} w_e$.

Why? For every cut $(T, \bar{T})$ it holds that

$$\sum_{\{i,j\} \in E : i \in T, j \in \bar{T}} w_e + \sum_{\{i,j\} \in E : i,j \in T \text{ or } i,j \in \bar{T}} w_e = \sum_{e \in E} w_e$$

Sketch of reduction

Let $\mathcal{I} = (G, w)$ be an instance of max-cut with FLIP neighborhood on graph $G = (V, E)$, with edge-weight function w.

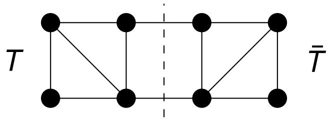
Maximizing weight of cut edges is equivalent to minimizing weight of non-cut edges (also locally).

Minimum uncut

Given undirected graph $G = (V, E)$ and weight function $w : E \to \mathbb{R}_{\geq 0}$, find partition $V = S \cup \bar{S}$ that minimizes $\sum_{\{i,j\} \in E : i,j \in S \text{ or } i,j \in \bar{S}} w_e$.

Why? For every cut $(T, \bar{T})$ it holds that

$$\sum_{\{i,j\} \in E : i \in T, j \in \bar{T}} w_e + \sum_{\{i,j\} \in E : i,j \in T \text{ or } i,j \in \bar{T}} w_e = \sum_{e \in E} w_e$$



Sketch of reduction

Let $\mathcal{I} = (G, w)$ be an instance of max-cut with FLIP neighborhood on graph $G = (V, E)$, with edge-weight function w.

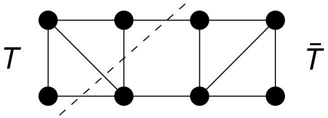
Maximizing weight of cut edges is equivalent to minimizing weight of non-cut edges (also locally).

Minimum uncut

Given undirected graph $G = (V, E)$ and weight function $w : E \to \mathbb{R}_{\geq 0}$, find partition $V = S \cup \bar{S}$ that minimizes $\sum_{\{i,j\} \in E : i,j \in S \text{ or } i,j \in \bar{S}} w_e$.

Why? For every cut $(T, \bar{T})$ it holds that

$$\sum_{\{i,j\} \in E : i \in T, j \in \bar{T}} w_e + \sum_{\{i,j\} \in E : i,j \in T \text{ or } i,j \in \bar{T}} w_e = \sum_{e \in E} w_e$$



We make a congestion games $\Gamma = (N, R, (S_i), (c_e))$ as follows:

We make a congestion games $\Gamma = (N, R, (S_i), (c_e))$ as follows:

- Nodes in V are the players N.

We make a congestion games $\Gamma = (N, R, (S_i), (c_e))$ as follows:

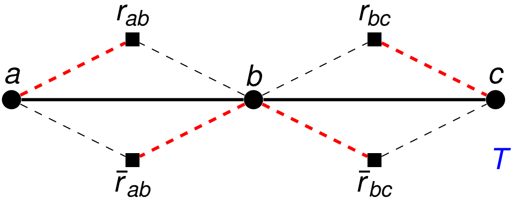
- Nodes in V are the players N.
- For $e \in E$, create two resources r_e and $\bar{r}_e$.

We make a congestion games $\Gamma = (N, R, (S_i), (c_e))$ as follows:

- Nodes in V are the players N.
- For $e \in E$, create two resources r_e and $\bar{r}_e$.
  - Let $R = e \cup_{e \in E} \{r_e, \bar{r}_e\}$.

We make a congestion games $\Gamma = (N, R, (S_i), (c_e))$ as follows:

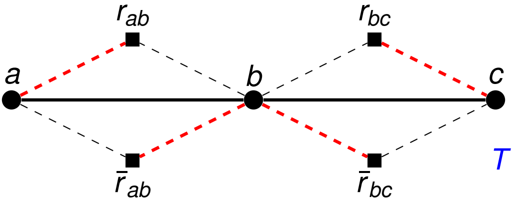
- Nodes in V are the players N.
- For $e \in E$, create two resources r_e and $\bar{r}_e$.
  - Let $R = e \cup_{e \in E} \{r_e, \bar{r}_e\}$.



$T = \{a, c\}, \bar{T} = \{b\}$

We make a congestion games $\Gamma = (N, R, (S_i), (c_e))$ as follows:

- Nodes in V are the players N.
- For $e \in E$, create two resources r_e and $\bar{r}_e$.
  - Let $R = e \cup_{e \in E} \{r_e, \bar{r}_e\}$.



$T = \{a, c\}, \bar{T} = \{b\}$

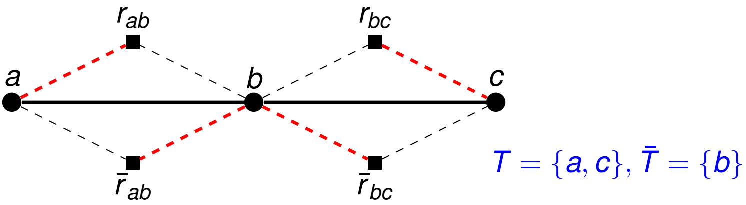
- Player $i \in V$ has two strategies ($\mathcal{S}_i = \{t_i, \bar{t}_i\}$):

$$t_i = \{r_e\}_{e \in \delta(i)} \quad \text{and} \quad \bar{t}_i = \{\bar{r}_e\}_{e \in \delta(i)}$$

where $\delta(i)$ is the set of all edges incident to i in G.

We make a congestion games $\Gamma = (N, R, (S_i), (c_e))$ as follows:

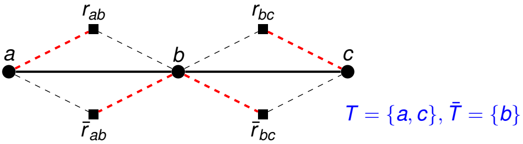
- Nodes in V are the players N.
- For $e \in E$, create two resources r_e and $\bar{r}_e$.
  - Let $R = e \cup_{e \in E} \{r_e, \bar{r}_e\}$.



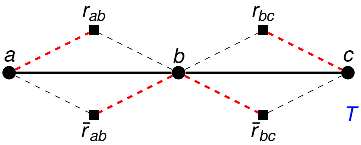
- Player $i \in V$ has two strategies ($\mathcal{S}_i = \{t_i, \bar{t}_i\}$):

$$t_i = \{r_e\}_{e \in \delta(i)} \quad \text{and} \quad \bar{t}_i = \{\bar{r}_e\}_{e \in \delta(i)}$$

  where $\delta(i)$ is the set of all edges incident to i in G.
  - These roughly model the choice between T and $\bar{T}$ for a node in V.

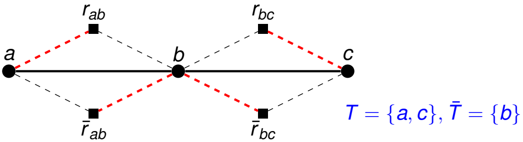


- Cost function for r_e (and $\bar{r}_e$) given by $c_{r_e}(1) = 0$ and $c_{r_e}(2) = \frac{w_e}{2}$.



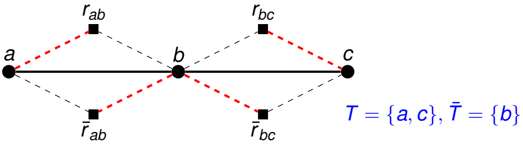
$T = \{a, c\}$, $\bar{T} = \{b\}$

- Cost function for r_e (and $\bar{r}_e$) given by $c_{r_e}(1) = 0$ and $c_{r_e}(2) = \frac{w_e}{2}$.
  - This is enough as at most two players can use the same resource.



- Cost function for r_e (and $\bar{r}_e$) given by $c_{r_e}(1) = 0$ and $c_{r_e}(2) = \frac{w_e}{2}$.
  - This is enough as at most two players can use the same resource.
  - For strategy profile $s = (s_1, \dots, s_n)$,

$$C_i(t_i, s_{-i}) = \frac{1}{2} \sum_{j \in \delta(i) : s_j = t_j} w_{ij} \quad \text{and} \quad C_i(\bar{t}_i, s_{-i}) = \frac{1}{2} \sum_{j \in \delta(i) : s_j = \bar{t}_j} w_{ij}.$$

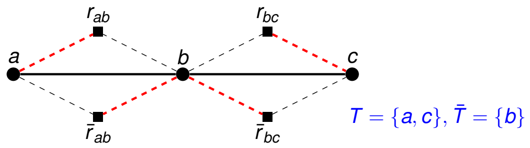


- Cost function for r_e (and $\bar{r}_e$) given by $c_{r_e}(1) = 0$ and $c_{r_e}(2) = \frac{w_e}{2}$.
  - This is enough as at most two players can use the same resource.
  - For strategy profile $s = (s_1, \ldots, s_n)$,

$$C_i(t_i, s_{-i}) = \frac{1}{2} \sum_{j \in \delta(i) : s_j = t_j} w_{ij} \quad \text{and} \quad C_i(\bar{t}_i, s_{-i}) = \frac{1}{2} \sum_{j \in \delta(i) : s_j = \bar{t}_j} w_{ij}.$$

Rosenthal's potential here is given by

$$\Phi(s) = \sum_{i \in V} C_i(s)$$



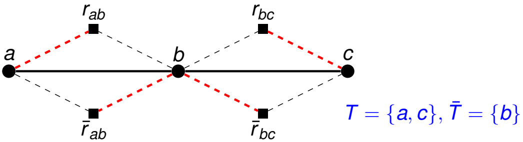
- Cost function for r_e (and $\bar{r}_e$) given by $c_{r_e}(1) = 0$ and $c_{r_e}(2) = \frac{w_e}{2}$.
  - This is enough as at most two players can use the same resource.
  - For strategy profile $s = (s_1, \ldots, s_n)$,

$$C_i(t_i, s_{-i}) = \frac{1}{2} \sum_{j \in \delta(i): s_j = t_j} w_{ij} \quad \text{and} \quad C_i(\bar{t}_i, s_{-i}) = \frac{1}{2} \sum_{j \in \delta(i): s_j = \bar{t}_j} w_{ij}.$$

Rosenthal's potential here is given by

$$\Phi(s) = \sum_{i \in V} C_i(s)$$

- Precisely the sum of non-cut edge weights!



- Cost function for r_e (and $\bar{r}_e$) given by $c_{r_e}(1) = 0$ and $c_{r_e}(2) = \frac{w_e}{2}$.
  - This is enough as at most two players can use the same resource.
  - For strategy profile $s = (s_1, \ldots, s_n)$,

$$C_i(t_i, s_{-i}) = \frac{1}{2} \sum_{j \in \delta(i): s_j = t_j} w_{ij} \quad \text{and} \quad C_i(\bar{t}_i, s_{-i}) = \frac{1}{2} \sum_{j \in \delta(i): s_j = \bar{t}_j} w_{ij}.$$

Rosenthal's potential here is given by

$$\Phi(s) = \textstyle\sum_{i \in V} C_i(s)$$

- Precisely the sum of non-cut edge weights!

PNEs of game are precisely locally min-uncuts/max-cuts!

Smoothed analysis (extra)

Smoothed analysis for local search

Smoothed analysis studies algorithmic problems under (small) perturbations of the input.

Smoothed analysis for local search

Smoothed analysis studies algorithmic problems under (small) perturbations of the input.

- Roughly speaking, to study if worst-case instances are rare or not.

Smoothed analysis for local search

Smoothed analysis studies algorithmic problems under (small) perturbations of the input.

- Roughly speaking, to study if worst-case instances are rare or not.

Max-cut with FLIP local search (informal)

For every $e \in E$, we introduce an (independent) random perturbation

$$\sigma_e \sim U[0, \phi],$$

where ϕ is a parameter,

Smoothed analysis for local search

Smoothed analysis studies algorithmic problems under (small) perturbations of the input.

- Roughly speaking, to study if worst-case instances are rare or not.

Max-cut with FLIP local search (informal)

For every $e \in E$, we introduce an (independent) random perturbation

$$\sigma_e \sim U[0, \phi],$$

where ϕ is a parameter, and focus on instance with perturbed weights

$$w'_e = w_e + \sigma_e.$$

Smoothed analysis for local search

Smoothed analysis studies algorithmic problems under (small) perturbations of the input.

- Roughly speaking, to study if worst-case instances are rare or not.

Max-cut with FLIP local search (informal)

For every $e \in E$, we introduce an (independent) random perturbation

$$\sigma_e \sim U[0, \phi],$$

where ϕ is a parameter, and focus on instance with perturbed weights

$$w'_e = w_e + \sigma_e.$$

Goal: *Show that every sequence of local improvements converges to a local optimum in time polynomial in n and ϕ (in perturbed instance).*

Smoothed analysis for local search

Smoothed analysis studies algorithmic problems under (small) perturbations of the input.

- Roughly speaking, to study if worst-case instances are rare or not.

Max-cut with FLIP local search (informal)

For every $e \in E$, we introduce an (independent) random perturbation

$$\sigma_e \sim U[0, \phi],$$

where ϕ is a parameter, and focus on instance with perturbed weights

$$w'_e = w_e + \sigma_e.$$

Goal: *Show that every sequence of local improvements converges to a local optimum in time polynomial in n and ϕ (in perturbed instance).*

- If $\phi \to \infty$, we get completely random instance.

Smoothed analysis for local search

Smoothed analysis studies algorithmic problems under (small) perturbations of the input.

- Roughly speaking, to study if worst-case instances are rare or not.

Max-cut with FLIP local search (informal)

For every $e \in E$, we introduce an (independent) random perturbation

$$\sigma_e \sim U[0, \phi],$$

where ϕ is a parameter, and focus on instance with perturbed weights

$$w_e' = w_e + \sigma_e.$$

Goal: *Show that every sequence of local improvements converges to a local optimum in time polynomial in n and ϕ (in perturbed instance).*

- If $\phi \to \infty$, we get completely random instance.
- If $\phi \to 0$, we get back (original) instance with weights w_e.

Smoothed analysis essentially interpolates between

Smoothed analysis essentially interpolates between

- Average-case analysis ($\phi \to \infty$);

Smoothed analysis essentially interpolates between

- Average-case analysis ($\phi \to \infty$);
- Worst-case analysis ($\phi \to 0$).

Smoothed analysis essentially interpolates between

- Average-case analysis ($\phi \to \infty$);
- Worst-case analysis ($\phi \to 0$).

What is known for max-cut in the literature?

Smoothed analysis essentially interpolates between

- Average-case analysis ($\phi \to \infty$);
- Worst-case analysis ($\phi \to 0$).

What is known for max-cut in the literature?

Theorem

Local search converges to a local optimum in at most

- $\phi n^{O(\log(n))}$ *steps for general graphs G;*
- *poly*(ϕ, n) *steps for complete graphs G;*
- *poly*(ϕ, n) *steps for graphs with* $\Delta(G) = O(\log(n))$.

Smoothed analysis essentially interpolates between

- Average-case analysis ($\phi \to \infty$);
- Worst-case analysis ($\phi \to 0$).

What is known for max-cut in the literature?

Theorem

Local search converges to a local optimum in at most

- $\phi n^{O(log(n))}$ *steps for general graphs G;*
- *poly*(ϕ, n) *steps for complete graphs G;*
- *poly*(ϕ, n) *steps for graphs with* $\Delta(G) = O(\log(n))$.

Big open question:

Smoothed analysis essentially interpolates between

- Average-case analysis ($\phi \to \infty$);
- Worst-case analysis ($\phi \to 0$).

What is known for max-cut in the literature?

Theorem

Local search converges to a local optimum in at most

- $\phi n^{O(\log(n))}$ *steps for general graphs G;*
- *poly*(ϕ, n) *steps for complete graphs G;*
- *poly*(ϕ, n) *steps for graphs with* $\Delta(G) = O(\log(n))$.

Big open question: Does (smoothed) local search for max-cut always converge in polynomial number of steps, for any graph G?