

Overview

Introduction

- Selfish Network Creation
- Model & Previous Work
- Summary of Results

Playing On Trees

- Analysing an Edge-Swap
- Max Cost Best Response Dynamic

General Graphs

- BRDs can cycle

Computing Best Responses

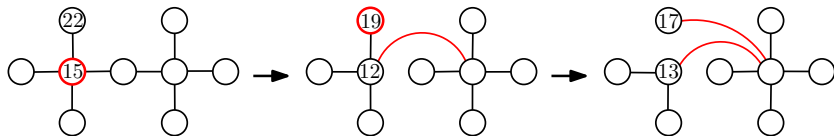
- On Trees
- In general graphs

Final Remarks

- Open Problems



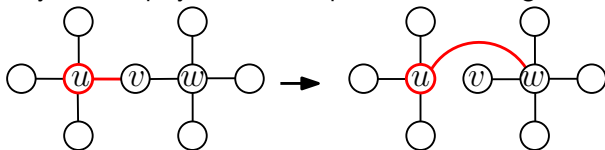
Selfish Network Creation



- n selfish players want to create a connected network G
- players can locally modify network structure:
 - actions: buy a new link, remove link, “swap” link
- each player wants to minimize her cost for network usage
 - SUM-measure: $c(v) = \sum_{w \in V} d(v, w)$
- *stable* network: no player can unilaterally decrease her costs by performing a move

Network Creation Models

- BUY-MODEL [Fabrikant et al., PODC'03]:
 - players can buy any incident edge for the price of $\alpha \geq 0$
 - players can remove incident edges
 - computing a best response is NP-hard
- SWAP-MODEL [Alon et al., SPAA'10]:
 - network G is given
 - only action: players can “swap” an incident edge for free



- swap models locally weighing decisions (possible edges)
- efficient computation of a best response

- Previous work focused on *static* properties (Price of Stability, Price of Anarchy, Structure) of equilibrium states
- ... but what about *dynamic* properties of these games?

Question:

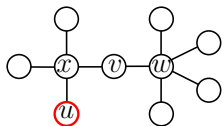
How can selfish and myopic players actually **find** such states?

- easiest version: finding stable states by *sequentially* choosing strategies
 - every move improving: *Improving Response Dynamic*
 - only optimal moves: *Best Response Dynamic*
- Do such greedy dynamics always converge?
- If yes - how long does it take until a stable graph emerges?
- Mechanism Design: Can this process be sped up by introducing coordination?

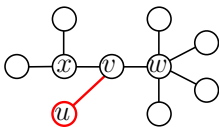
Our Contribution for the SWAP-MODEL

- When played on a tree:
 - Ordinal Potential Game
 - ⇒ any improving response dynamic converges to a star
 - $\mathcal{O}(n^3)$ steps needed for convergence
 - almost optimal speed up to $\approx \frac{3}{2}n$ steps if best responses are played and high cost players are favored
 - $\mathcal{O}(n)$ algorithm for computing a best response, even if $k > 1$ edges can be swapped at a time
- On general graphs:
 - best response dynamics can cycle
 - ⇒ fundamentally different techniques are needed for analysis
 - computation of best response NP-hard if $k > 1$ edges can be swapped at a time - even on very simple general graphs

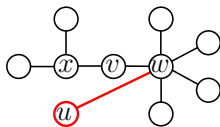
Playing on a Tree



$$c(u) = 26$$



$$c(u) = 23$$

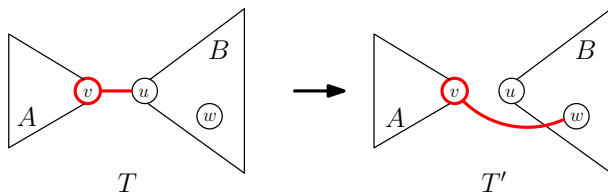


$$c(u) = 22$$

- Tree $T = (V, E)$ is given
- Cost of player v is $c(v) = \sum_{w \in V} d(v, w)$
- Social Cost of T is $\Phi_T = \sum_{v \in V} c(v)$
- player u is called *active* if u can swap to strictly decrease cost
- any swap that strictly decreases a player's cost is called an *improving response*
- any swap that decreases a player's cost most is called a *best response*



Impact of an Edge-Swap



- for any subtree K and player z : let $c_K(z) = \sum_{x \in K} d(x, z)$
- let $|K|$ denote the number of vertices in subtree K
- player v performs the swap vu to vw :
 - v 's distance to players in A does not change
 - v 's distance to players in B can change:

$$c_B(v) = \sum_{x \in B} (1 + d(u, x)) \quad \text{to} \quad c'_B(v) = \sum_{x \in B} (1 + d(w, x))$$

$$\Rightarrow v\text{'s change: } \Delta(v) = c_B(u) - c_B(w)$$

$$\Rightarrow \text{change in social cost: } \Delta\Phi = \Phi_T - \Phi_{T'} = 2|A|\Delta(v)$$



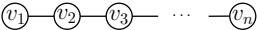
Ordinal Potential Game & IRDs

- change in social cost: $\Delta\Phi = 2|A|\Delta(v)$
 - $\Rightarrow \Phi$ decreases if and only if player v 's cost decreases
 - $\Rightarrow \Phi$ is an *ordinal potential function*!
 - Ordinal Potential Games:
 - Pure Nash Equilibria (=stable graphs) always exist
 - selfish and myopic behavior leads to convergence towards PNE
 - the only stable tree is a star [Alon et al.]
- \Rightarrow every Improving Response Dynamic on trees converges to a star

Question

How many steps are needed for convergence?

Convergence of IRDs on Trees

- $s(n)$: number of steps needed for convergence by an IRD on a tree T with n vertices
- P_n :  $(v_1) - (v_2) - (v_3) - \dots - (v_n)$
- lower bound: $s(n) \geq n - 3$
Idea: For $n \geq 4$, inner vertex of P_n has $n - 3$ non-neighbors
 \Rightarrow at least $n - 3$ swaps needed until star emerges
- upper bound: $s(n) \leq \frac{n^3}{6} - n^2 + \frac{11n}{6} - 1 \in \mathcal{O}(n^3)$
Idea: Start with max-cost tree (which is P_n), assume least possible Φ -decrease per step (which is 2)

Faster Convergence

- Goal: Speeding up convergence by imposing constraints on the dynamic
 - better moves: enforce Best Response Dynamic
 - fairness: player with highest cost is allowed to move

Max Cost Best Response Dynamic

In every step, an active player having the highest cost is allowed to play a best response. (Break ties arbitrarily.)

- $s^*(n)$: number of steps needed for convergence by mcBRD on a tree having n vertices

Theorem

- $s^*(n) \leq n - 3$, if n is even
- $s^*(n) \leq n + \lfloor n/2 \rfloor - 5$, if n is odd



Analysing mcBRD

- A vertex having minimum cost is called a *center-vertex*.

Some observations:

- (1) Only leaves move.
- (2) A best response move of leaf l is to swap towards a center-vertex in tree $T - l$.
- (3) A tree can have at most two center-vertices.
- (4) If n is odd, then the center-vertex is unique.
- (5) If the moving player of step i has a unique best response move towards vertex w , then all players who move in a later step will swap towards w .

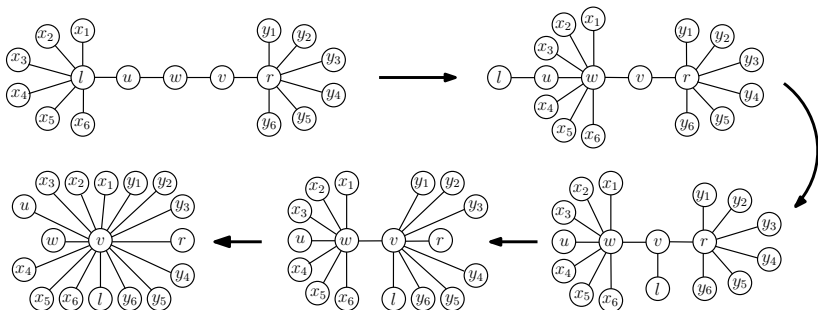
Proof of Theorem for even n .

- by (1), first move is made by a leaf l
- $T - l$ is a tree having an odd number of vertices
 \Rightarrow by (4), $T - l$ has a unique center-vertex w
- by (2), player l 's best move is to swap towards w
- by (5), since l had a unique best response, all players moving in a later step will swap towards w
- by (5) and (2), every leaf already connected to w will never move again
 - \Rightarrow every player moves at most once
 - \Rightarrow all non-neighbors of w will move exactly once
 - \Rightarrow at most $n - 3$ steps, since $\deg(w) \geq 2$



- Analysis of mcBRD for odd n is tight: There is a family of trees, where mcBRD can take $n + \lfloor n/2 \rfloor - 5$ steps.

Example for $n = 17$, mcBRD takes $n + \lfloor n/2 \rfloor - 5 = 20$ steps:

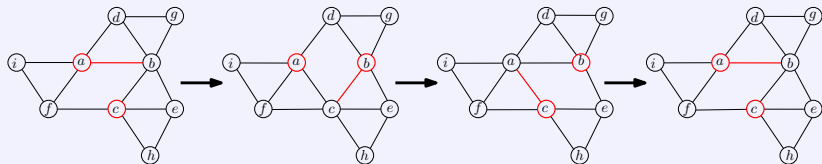


Best Response Dynamics on General Graphs

Theorem

Best Response Dynamics on general graphs can cycle.

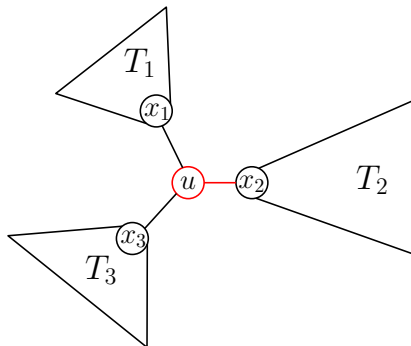
Proof.



- ⇒ there cannot exist a potential function for the SWAP-MODEL on general graphs
- ⇒ no (easy) guarantee for convergence to a PNE

Computing Best Responses on Trees

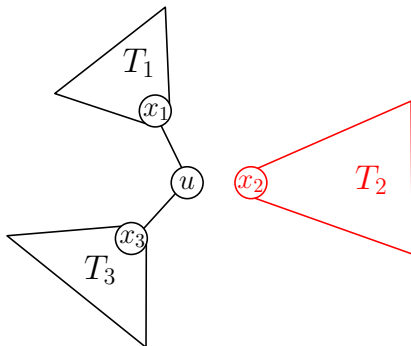
- How to compute player u 's best response if u can swap k edges at a time?
 - ⇒ reduces to computing center-vertices of subtrees



consider edge ux_2

Computing Best Responses on Trees

- How to compute player u 's best response if u can swap k edges at a time?
 - ⇒ reduces to computing center-vertices of subtrees

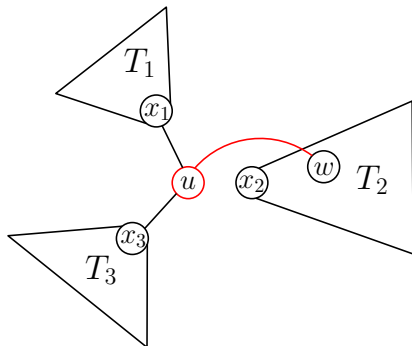


Removal of ux_2 disconnects T

⇒ u must create edge uw , with $w \in T_2$

Computing Best Responses on Trees

- How to compute player u 's best response if u can swap k edges at a time?
 - \Rightarrow reduces to computing center-vertices of subtrees



Which vertex in T_2 is optimal?

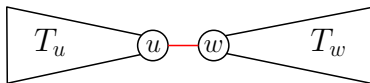
\Rightarrow choose w such that

$$w \in \arg \min_{z \in T_2} c_{T_2}(z)$$

$\Rightarrow w$ is a *center-vertex* in T_2 !

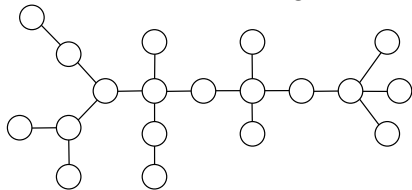
Computing Best Responses on Trees

- Given a tree T , how to compute a center-vertex of T ?
 - naive: compute $c(v)$ for all $v \in T \Rightarrow \mathcal{O}(|T|^2)$ steps
 - better: use the following observation:



$$c(u) \leq c(w) \iff |T_u| \geq |T_w|$$

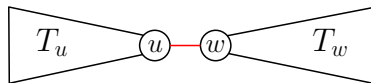
\Rightarrow linear time algorithm:



- start from leaves, compute $|K|$, where K is a (union of) already processed subtrees

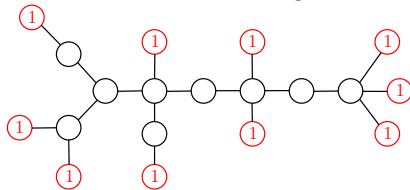
Computing Best Responses on Trees

- Given a tree T , how to compute a center-vertex of T ?
 - naive: compute $c(v)$ for all $v \in T \Rightarrow \mathcal{O}(|T|^2)$ steps
 - better: use the following observation:



$$c(u) \leq c(w) \iff |T_u| \geq |T_w|$$

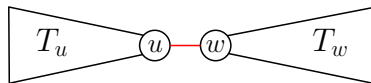
\Rightarrow linear time algorithm:



- start from leaves, compute $|K|$, where K is a (union of) already processed subtrees

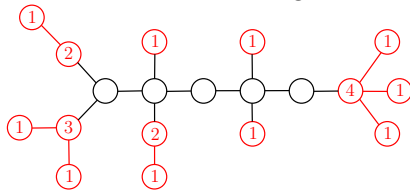
Computing Best Responses on Trees

- Given a tree T , how to compute a center-vertex of T ?
 - naive: compute $c(v)$ for all $v \in T \Rightarrow \mathcal{O}(|T|^2)$ steps
 - better: use the following observation:



$$c(u) \leq c(w) \iff |T_u| \geq |T_w|$$

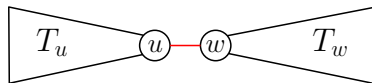
\Rightarrow linear time algorithm:



- start from leaves, compute $|K|$, where K is a (union of) already processed subtrees

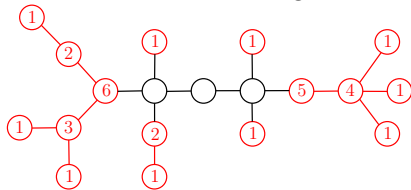
Computing Best Responses on Trees

- Given a tree T , how to compute a center-vertex of T ?
 - naive: compute $c(v)$ for all $v \in T \Rightarrow \mathcal{O}(|T|^2)$ steps
 - better: use the following observation:



$$c(u) \leq c(w) \iff |T_u| \geq |T_w|$$

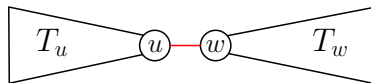
\Rightarrow linear time algorithm:



- start from leaves, compute $|K|$, where K is a (union of) already processed subtrees

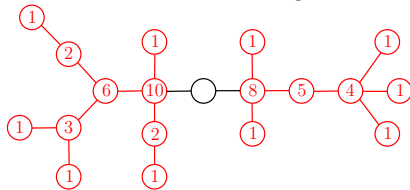
Computing Best Responses on Trees

- Given a tree T , how to compute a center-vertex of T ?
 - naive: compute $c(v)$ for all $v \in T \Rightarrow \mathcal{O}(|T|^2)$ steps
 - better: use the following observation:



$$c(u) \leq c(w) \iff |T_u| \geq |T_w|$$

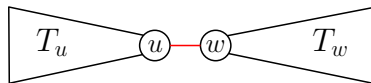
\Rightarrow linear time algorithm:



- start from leaves, compute $|K|$, where K is a (union of) already processed subtrees

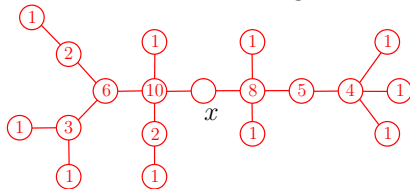
Computing Best Responses on Trees

- Given a tree T , how to compute a center-vertex of T ?
 - naive: compute $c(v)$ for all $v \in T \Rightarrow \mathcal{O}(|T|^2)$ steps
 - better: use the following observation:



$$c(u) \leq c(w) \iff |T_u| \geq |T_w|$$

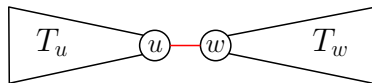
\Rightarrow linear time algorithm:



- start from leaves, compute $|K|$, where K is a (union of) already processed subtrees
- \Rightarrow process terminates at vertex x

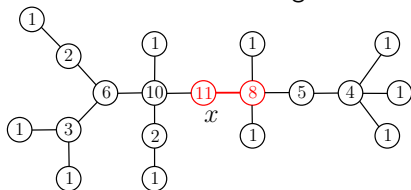
Computing Best Responses on Trees

- Given a tree T , how to compute a center-vertex of T ?
 - naive: compute $c(v)$ for all $v \in T \Rightarrow \mathcal{O}(|T|^2)$ steps
 - better: use the following observation:



$$c(u) \leq c(w) \iff |T_u| \geq |T_w|$$

\Rightarrow linear time algorithm:



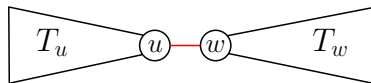
- start from leaves, compute $|K|$, where K is a (union of) already processed subtrees

\Rightarrow process terminates at vertex x

- use observation to perform local search starting from x

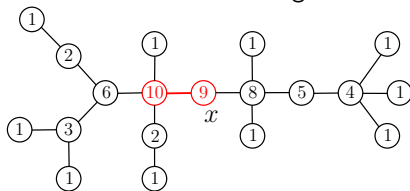
Computing Best Responses on Trees

- Given a tree T , how to compute a center-vertex of T ?
 - naive: compute $c(v)$ for all $v \in T \Rightarrow \mathcal{O}(|T|^2)$ steps
 - better: use the following observation:



$$c(u) \leq c(w) \iff |T_u| \geq |T_w|$$

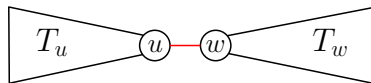
\Rightarrow linear time algorithm:



- start from leaves, compute $|K|$, where K is a (union of) already processed subtrees
- \Rightarrow process terminates at vertex x
- use observation to perform local search starting from x

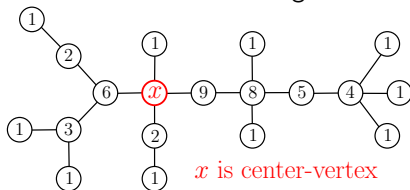
Computing Best Responses on Trees

- Given a tree T , how to compute a center-vertex of T ?
 - naive: compute $c(v)$ for all $v \in T \Rightarrow \mathcal{O}(|T|^2)$ steps
 - better: use the following observation:



$$c(u) \leq c(w) \iff |T_u| \geq |T_w|$$

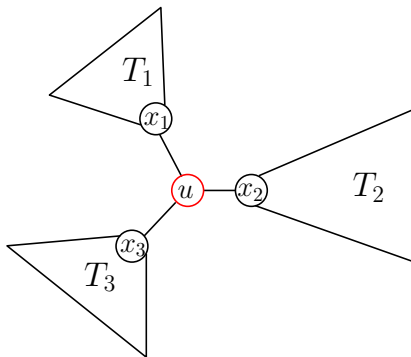
\Rightarrow linear time algorithm:



- start from leaves, compute $|K|$, where K is a (union of) already processed subtrees
- \Rightarrow process terminates at vertex x
- use observation to perform local search starting from x

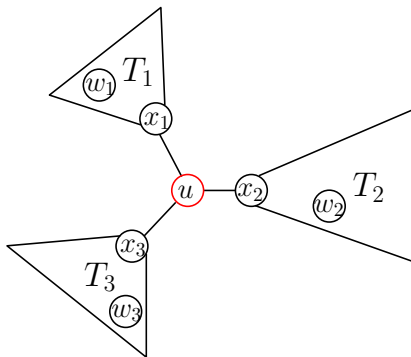
Computing Best Responses on Trees

- How to compute player u 's best response if u can swap k edges at a time?
⇒ $\mathcal{O}(n)$ Algorithm:



Computing Best Responses on Trees

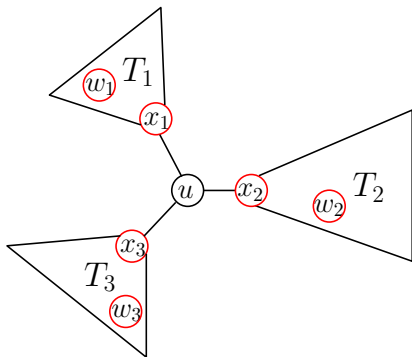
- How to compute player u 's best response if u can swap k edges at a time?
 $\Rightarrow \mathcal{O}(n)$ Algorithm:



- compute center-vertex w_i for every tree $T_i \Rightarrow \mathcal{O}(n)$ steps

Computing Best Responses on Trees

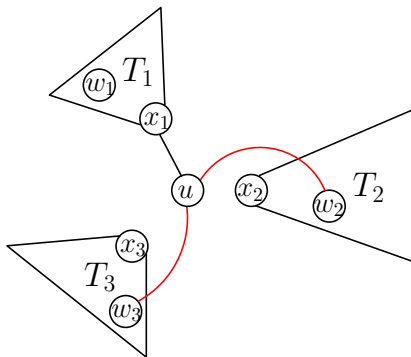
- How to compute player u 's best response if u can swap k edges at a time?
 $\Rightarrow \mathcal{O}(n)$ Algorithm:



- compute center-vertex w_i for every tree $T_i \Rightarrow \mathcal{O}(n)$ steps
- compute $d_i = c_{T_i}(x_i) - c_{T_i}(w_i)$ for every tree $T_i \Rightarrow \mathcal{O}(n)$ steps

Computing Best Responses on Trees

- How to compute player u 's best response if u can swap k edges at a time?
 $\Rightarrow \mathcal{O}(n)$ Algorithm:



- compute center-vertex w_i for every tree $T_i \Rightarrow \mathcal{O}(n)$ steps
- compute $d_i = c_{T_i}(x_i) - c_{T_i}(w_i)$ for every tree $T_i \Rightarrow \mathcal{O}(n)$ steps
- performing the $\min\{k, \deg(u)\}$ most profitable swaps is player u 's best response (use radix sort)
 $\Rightarrow \mathcal{O}(n)$ steps

Computing Best Responses in General Graphs

Question: Given a graph G , how hard is it to compute player u 's best response if u is allowed to swap k edges at a time?

- $k = 1$: try all possibilities $\Rightarrow \mathcal{O}(n^2)$ steps
- $k > 1$ swaps at a time:

Theorem

If players are allowed to swap $k > 1$ edges at a time, then computing the best response is NP-hard even if G is planar and has maximum degree 3.

Proofidea:

Reduction from the k -MEDIAN-PROBLEM.

Open Problems

Open Problem 1

If only best responses are played, how many steps are needed for convergence on trees?

Open Problem 2

Do restricted Best Response Dynamics (e.g. mcBRD) enforce convergence in general graphs? If yes - how fast?

