Algorithm Design

Algorithmic Game Theory

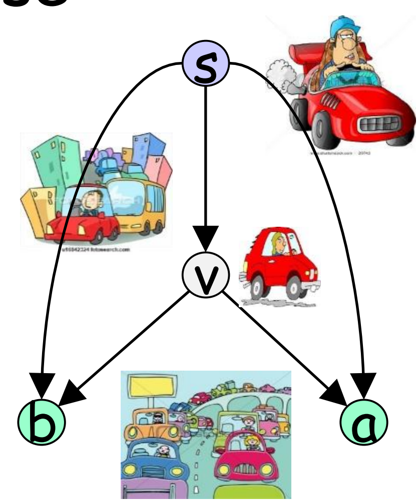
Part II

Congestion Games

Multicast Routing Cost Sharing: A game with **positive** congestion effect – players want to share resources with other players.

Network congestion: A game with **negative** congestion effect – players want to use resources with no (few) partners.

Both are justified by real applications.



General Congestion Game

- A congestion game is defined by a tuple

$\{N, M, \{A_i\}$ for all $i \in N$, $\{c_j\}$ for all $j \in M\}$

- $N = \{1..n\}$ denotes the set of players.
- $M = \{1..m\}$ denotes the set of resources.
- For $i \in N$, A_i denotes the set of strategies of player i, where each $a_i \in A_i$ is a non empty subset of the resources.
- For $j \in M$, $c_j \in R^n$ denotes the vector of costs, where $c_j(k)$ is the cost related to each user of resource j, if there are exactly k players using it

Example: Network Congestion Game

- A network congestion game is defined by a tuple

{N, a graph $G=(V,E)$, $\{P_i\}$ for all $i \in N$, $\{c_j\}$ for all $j \in M$}

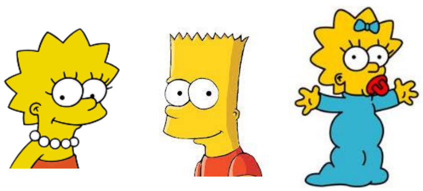
- $N = \{1..n\}$ denotes the set of players. Each player is associated with a source $s_i \in V$ and a target $t_i \in V$.
- $E = \{1..m\}$ denotes the graph edges
- For $i \in N$, P_i denotes the set of $(s_i\text{-}t_i)$-paths in G.
- For $j \in E$, $c_j \in R^n$ denotes the vector of costs, where $c_j(k)$ is the delay travelling on edge j when used by k players.

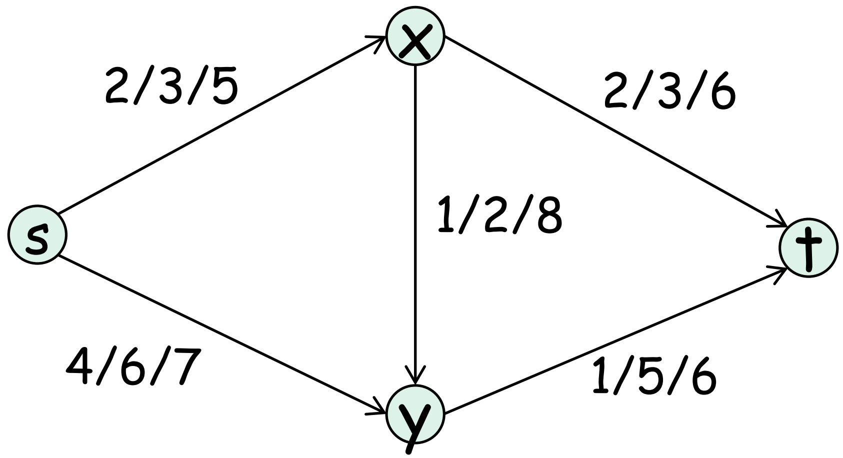
Example: Network Congestion Game

- A profile a is a set of strategies selected by the players.
- $a=(a_1, a_2,\ldots, a_n)\in(A_1 \times A_2 \times\ldots \times A_n)$
- Let $n_j(a)$ denote the number of players for which resource j is used in profile a. ($j \in a_i$)
- The cost function for player i in the profile a is:
  $u_i(a) = \sum_{j\in a_i} c_j\left(n_j(a)\right)$.
- Players are selfish.

Remark: All players are equal in a sense that they have the same 'weight' (it doesn't matter which players are using a facility, only how many players are using it).

Example: Network Congestion Game

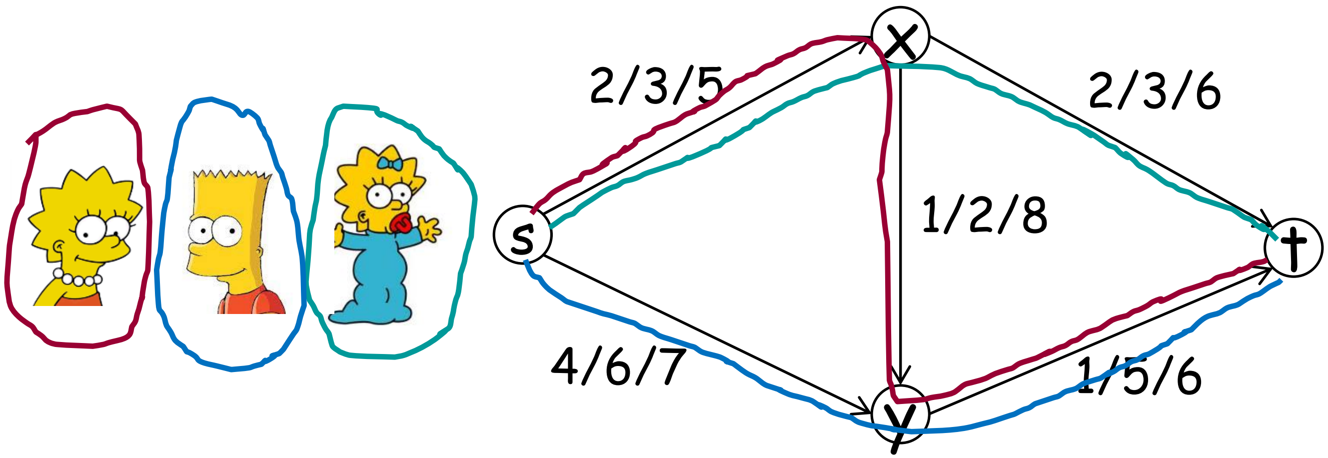




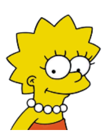
- Assume that players A,B,C have to go from s to t
- Edges are labeled by the function c_j.

For example, if two players are using the edge (yt) then each player experiences a delay of 5 on this edge.
The strategy set of all three players includes three paths: for all i=A,B,C P_i={{sx,xt},{sy,yt},{sx,xy,yt}}

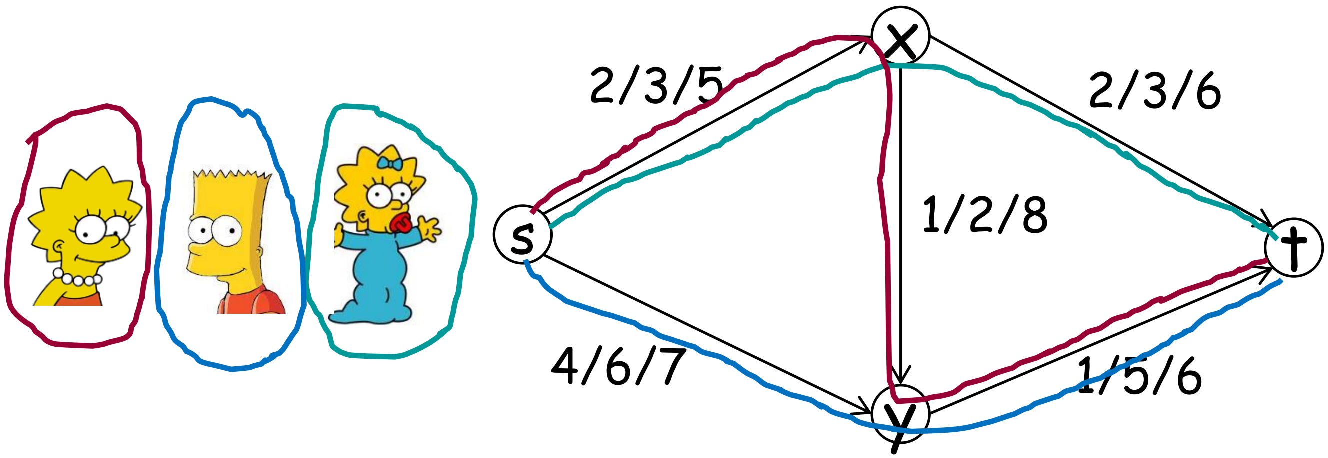
Network Congestion Game



Example:

-  selects the path s-x-t. cost (delay) = 3+2=5
-  selects the path s-x-y-t. cost = 3+1+5=9
-  selects the path s-y-t. cost = 4+5=9

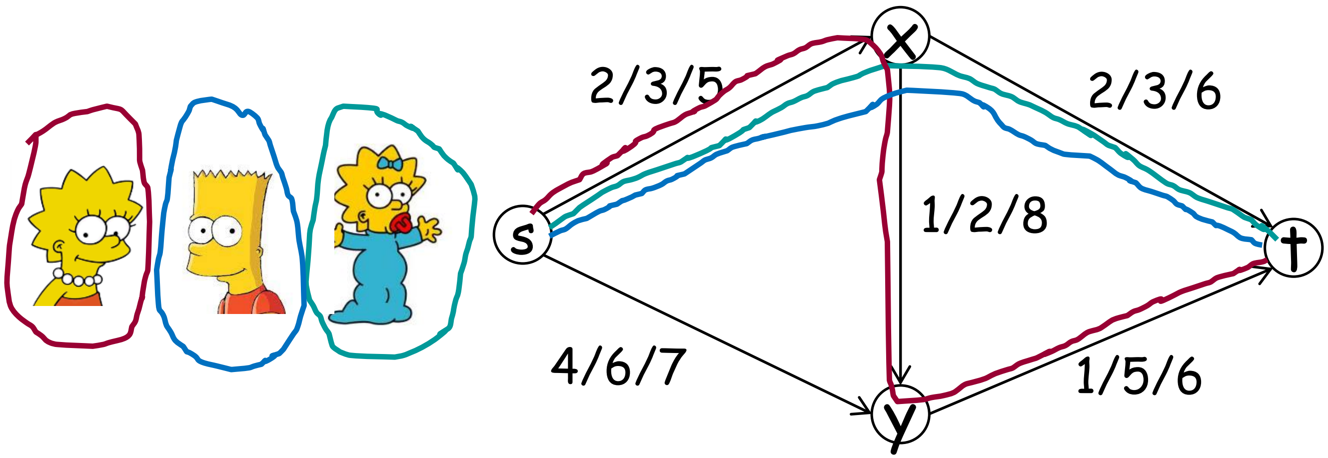
Network Congestion Game



Is it a Nash Equilibrium profile?

- Should  switch from s-y-t to s-x-t?
- Currently his cost is 9. By migrating…

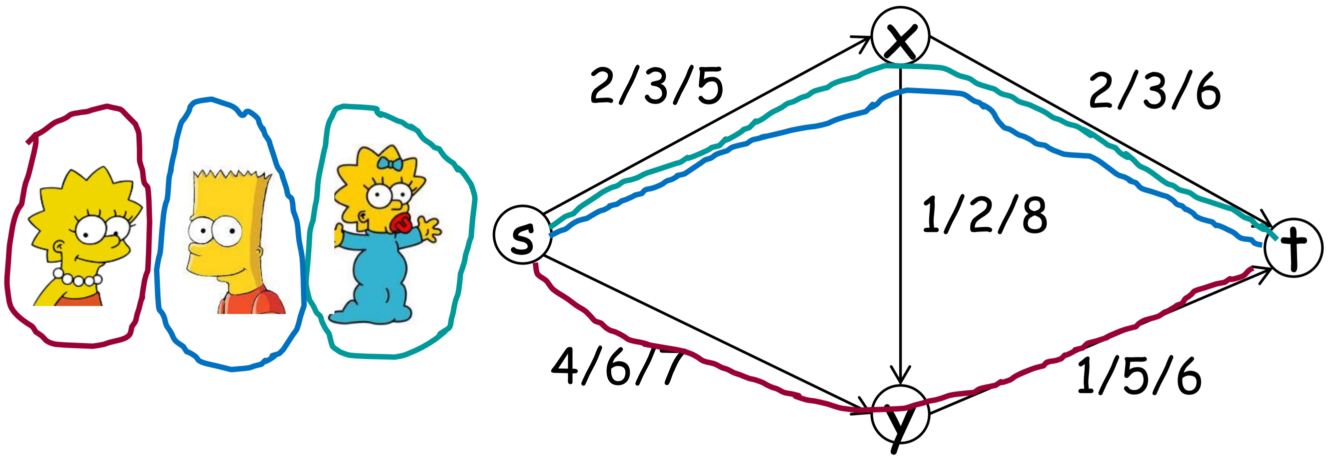
Network Congestion Game



Is it a Nash Equilibrium profile?

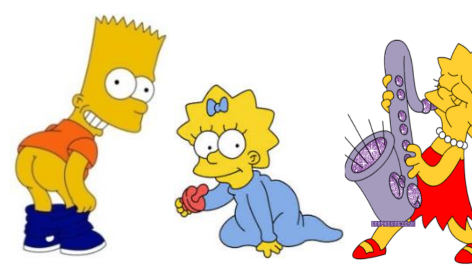
- Should  switch from s-y-t to s-x-t?
- Currently his cost is 9. By migrating his cost would reduce to 5+3=8.

Network Congestion Game

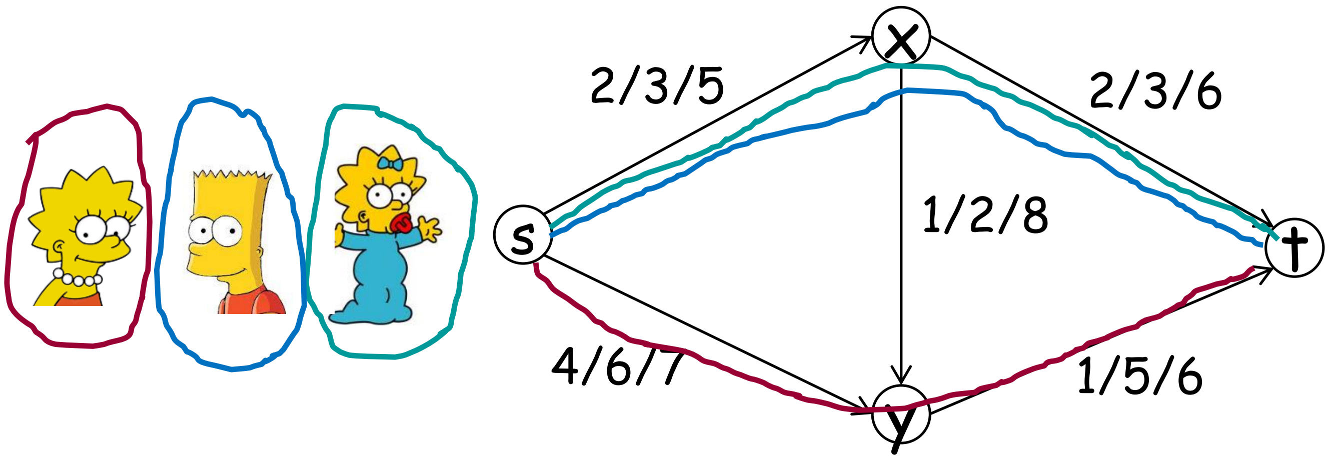


- Next,  is migrating and reduces her cost from 8 to 5.
- Each of   pays 6.

We have reached a NE.



Network Congestion Game



Note: Even though the game is symmetric (all players have the same objective), they do not share the same strategy in the NE.

Nash Equilibrium Existence.

Theorem: Every finite congestion game has a pure strategy Nash equilibrium.

Proof: Let a be a deterministic strategy vector as defined above, let $\Phi: A \rightarrow R$ be a potential function defined as follows:

$$\Phi(a) = \sum_{j=1}^{m} \sum_{k=1}^{n_j(a)} c_j(k)$$

Claim: For every improvement step of player i, it holds that $\Delta\Phi = \Delta u_i$.

Proof: in class.

Nash Equilibrium Existence.

Corollary: Since Φ can accept a finite number of values, the following algorithm (BRD) converges to a NE.

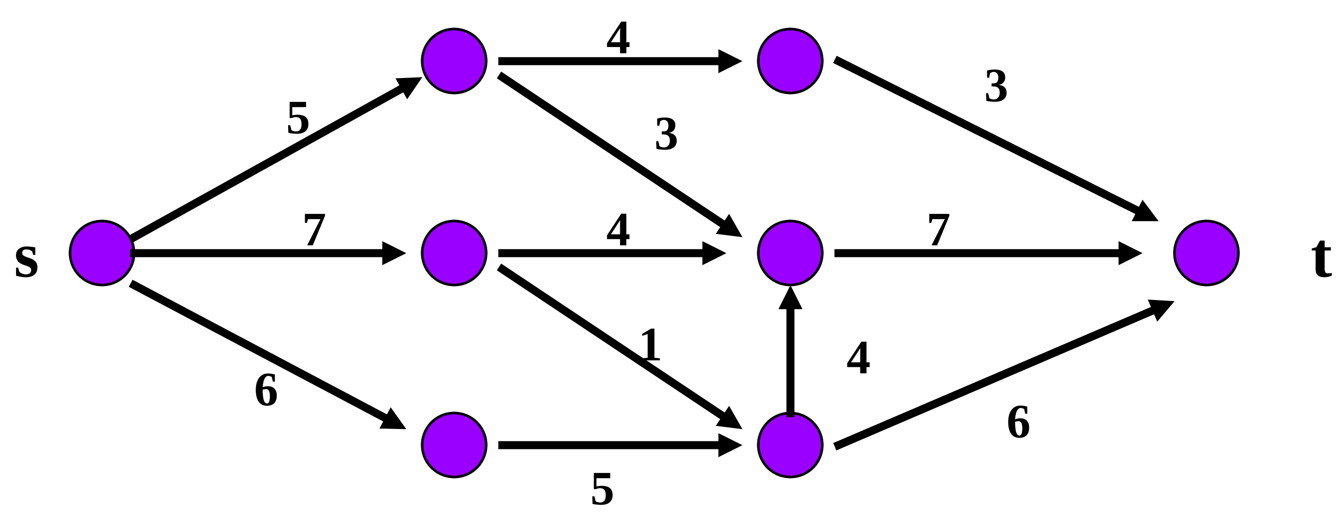
```
Best-Response-Dynamics(G,c) {
   Pick a path for each agent

   while (not a Nash equilibrium) {
      Pick an agent i who can improve by
       switching paths
      Switch path of agent i
   }
}
```

Computing a NE in congestion games

- For general congestion games, the problem of finding a NE is PLS-complete (PLS = Polynomial-time Local Search). Probably can't be solved in polynomial time).
- We will see a polynomial time algorithm for symmetric network congestion games.
- A network congestion game is symmetric is all the players have the same set of strategies (common source and target vertices).
- The algorithm is based on a reduction to a max-flow min-cost problem.

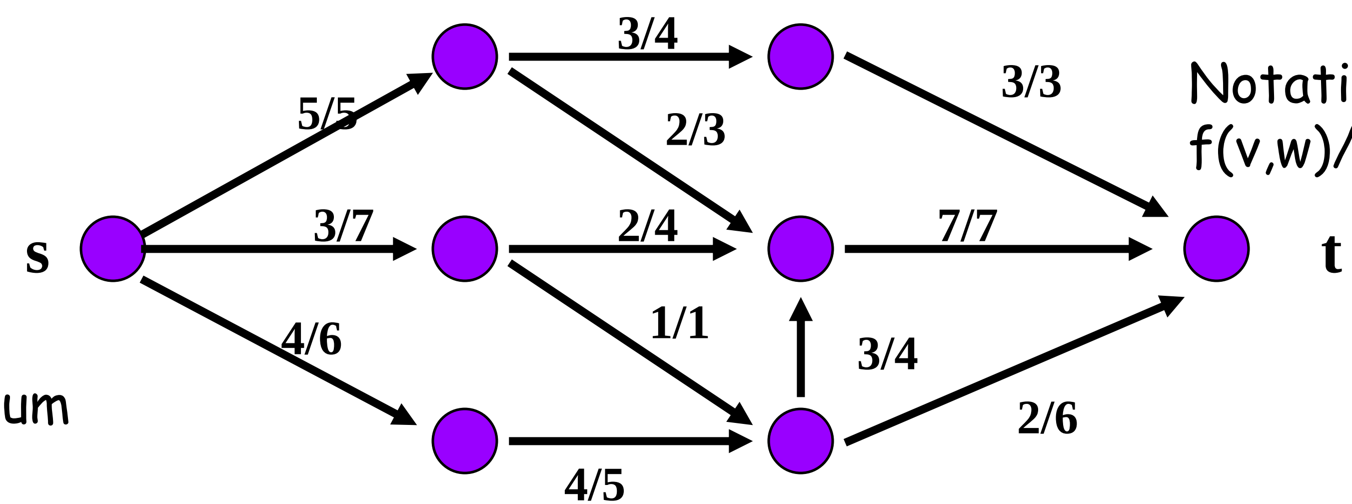
Maximum Flow (no costs)

- Input: a directed graph (network) G
  - each edge (v,w) has associated capacity $c(v,w)$
  - a specified source node s and target node t
- Problem: What is the maximum flow you can route from s to t while respecting the capacity constraint of each edge?



Properties of Flow: $f(v,w)$ - flow on edge (v,w)

- **Edge condition:** $0 \leq f(v,w) \leq c(v,w)$: the flow through an edge cannot exceed the capacity of an edge.
- **Vertex condition:** for all v except s,t :
  $\Sigma_u f(u,v) = \Sigma_w f(v,w)$: the total flow entering a vertex is equal to total flow exiting this vertex.
- total flow leaving s = total flow entering t.

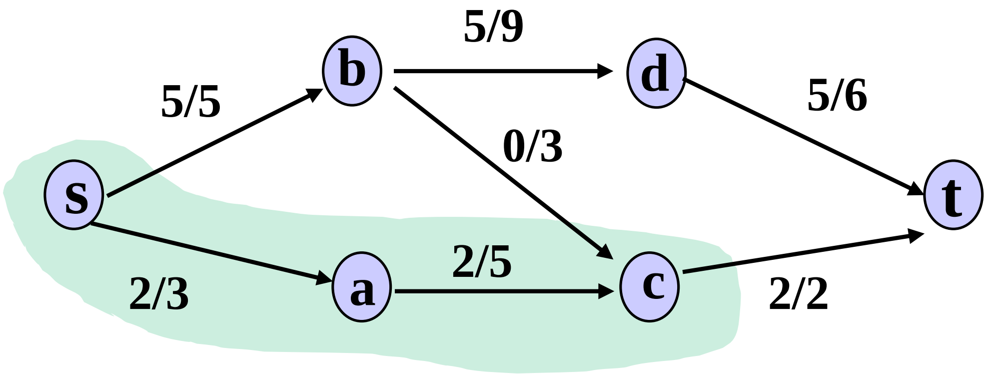


Notation on edges $f(v,w)/c(v,w)$

Not a maximum flow!

Max-flow Min-Cut Theorem

The value of a maximum flow in a network is equal to the minimum capacity of a cut.



Example: Flow value = max-cut capacity = 7

Back to NE calculation: max-flow min-cost

Given a flow-network G where each edge (v,w) has associated capacity $c(v,w)$, and **a cost** $cost(v,w)$.

The goal is to find a maximum flow of minimum cost.

The cost of a flow f : $\Sigma_{f(v,w)>0}\, cost(v,w) f(v,w)$

Out of all the maximum flows, which has minimal cost?

The max-flow min-cost problem has a polynomial time algorithm.

Computing a NE in a symmetric network congestion game

Input:

- A graph $G=(V,E)$
- A source $s \in V$ and a target $t \in V$.
- For each edge $j=1..m$, the cost function $c_j(k)$
- The number of players, n.

Output: A NE profile.

Computing a NE in a symmetric network congestion game

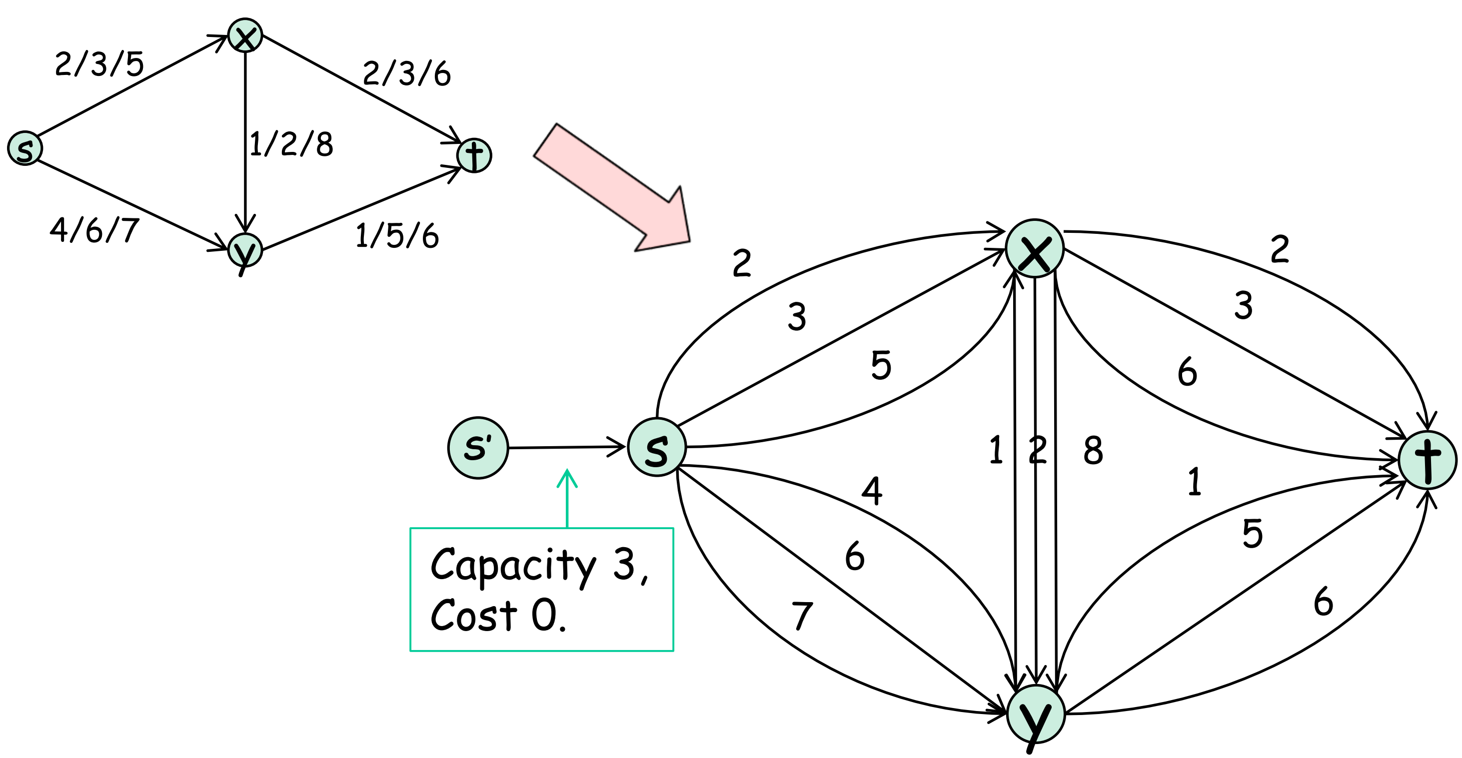
Algorithm:

- Build a flow network with costs: replace in G every edge e by n parallel edges between the same nodes, each with capacity 1, and with costs $c_e(1),\ldots, c_e(n)$.
- Find a min-cost flow of value n (how?)
- The flow induces n disjoint paths from s to t. These paths define a NE profile.

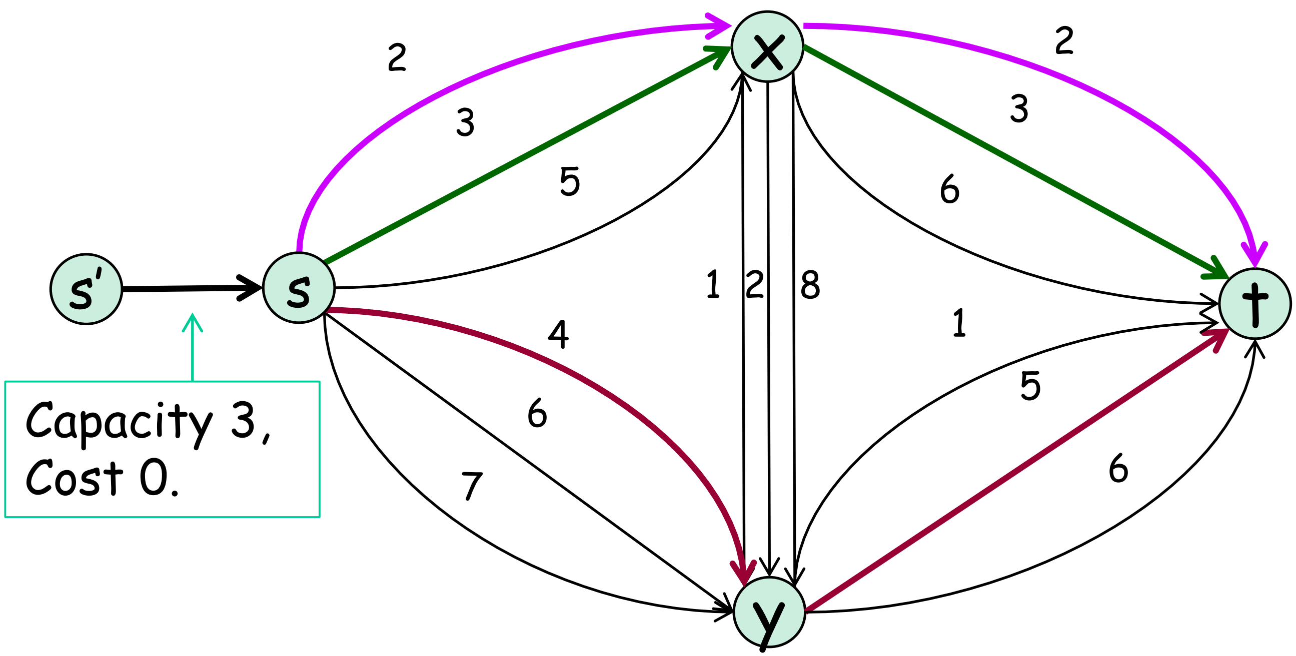
Proof and Analysis: In class

Computing a NE in a symmetric network congestion game

Construction example: edges are labeled by their costs, all edges have capacity 1.



Computing a NE in a symmetric network congestion game



Min-cost flow of value 3.

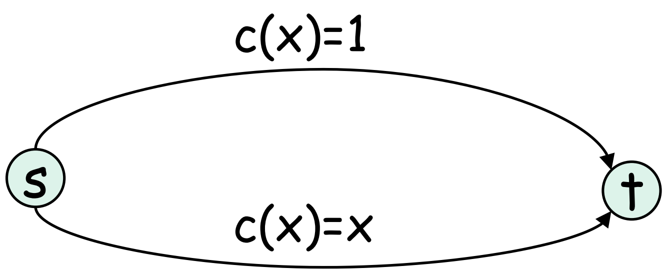
Computing a NE in a symmetric network congestion game

Note: The algorithm does not find a social optimum profile.

Minimizing the potential is not equal to minimizing the total players' cost!

How bad is selfish routing?

- Pigou's Network: The price of anarchy as well as the price of stability can be 4/3.

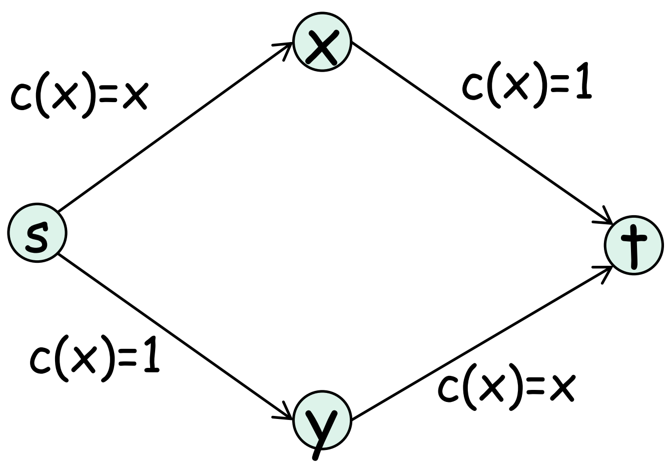


- For simplicity we assume the total load is 1 and measure the load on each edge by fractions (splittable flow).

Objective function: Minimize the average delay.
Social Optimum: split the flow, $\frac{1}{2}$ on top path, $\frac{1}{2}$ on lower path. Average delay is $(1 + \frac{1}{2})/2 = \frac{3}{4}$.
Unique NE: All players in lower path. Average delay is 1.

How bad is selfish routing?

- Braess's Paradox: The addition of an intuitively helpful link can negatively impact all of the users of a congested network.

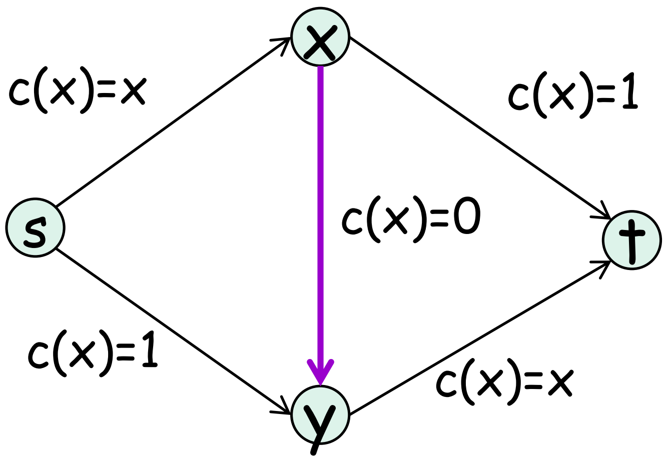


- Again, for simplicity we measure the load in fractions.
- What is the unique NE for routing of one unit of flow from s to t?

Answer: split the flow, ½ on top path, ½ on lower path.
The delay for all players is 3/2.
This NE is also the optimal social cost.

How bad is selfish routing?

- Suppose that a new edge with delay 0 (independent of the load) is added.



The optimal flow remains the same. The new edge is not used.

- However, it is not a NE!
- The new unique NE is when all the flow route in the path s-x-y-t. The corresponding delay is 2.

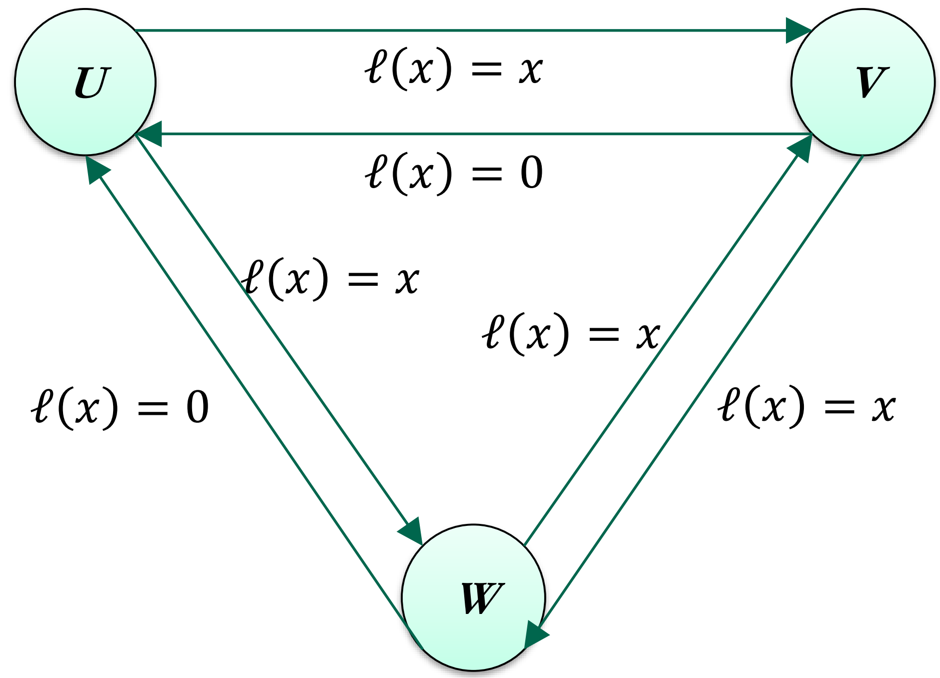
How bad is selfish routing?

Known results:

- If the delay function of each edge is a linear function of the edge congestion then the price of anarchy is at most 4/3, and this is tight.
- If the delay functions assumed only to be non-decreasing, then the price of anarchy is unbounded.
- Many results for specific cost functions or specific network structure.
- Analysis of static one-shot flow.

Example: Unsplittable Flow

Consider the following network:

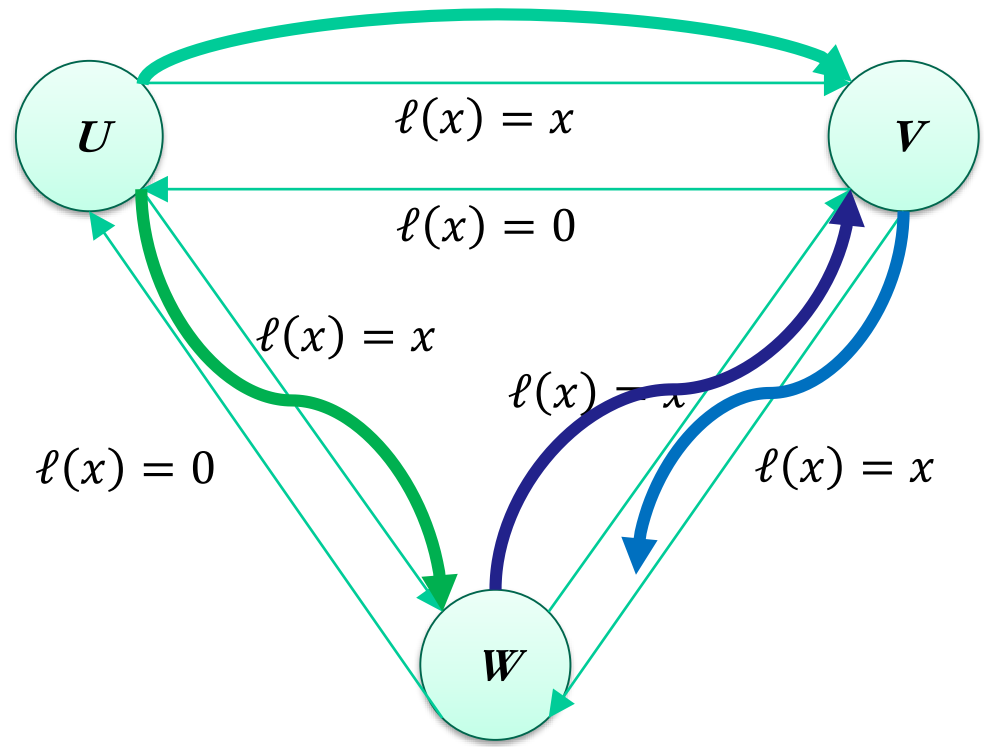


- There are 4 players with the following requests:

  (s_i, t_i): $\{(U, V); (U, W); (V, W); (W, V)\}$

Calculating OPT's Cost

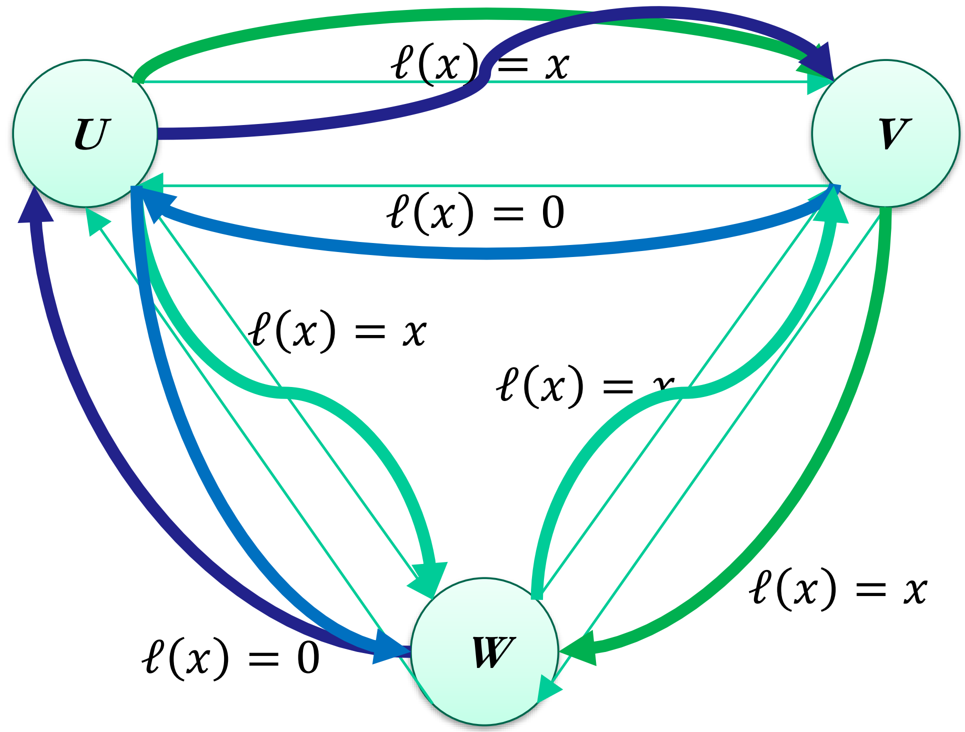
- OPT routes:
  - $(s_1, t_1) = (U, V)$
  - $(s_2, t_2) = (U, W)$
  - $(s_3, t_3) = (V, W)$
  - $(s_4, t_4) = (W, V)$



- Total Cost $= 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 = 4$

Calculating NE's Cost and PoA

- A possible NE
  - $(s_1, t_1) = (U, V)$
  - $(s_2, t_2) = (U, W)$
  - $(s_3, t_3) = (V, W)$
  - $(s_4, t_4) = (W, V)$

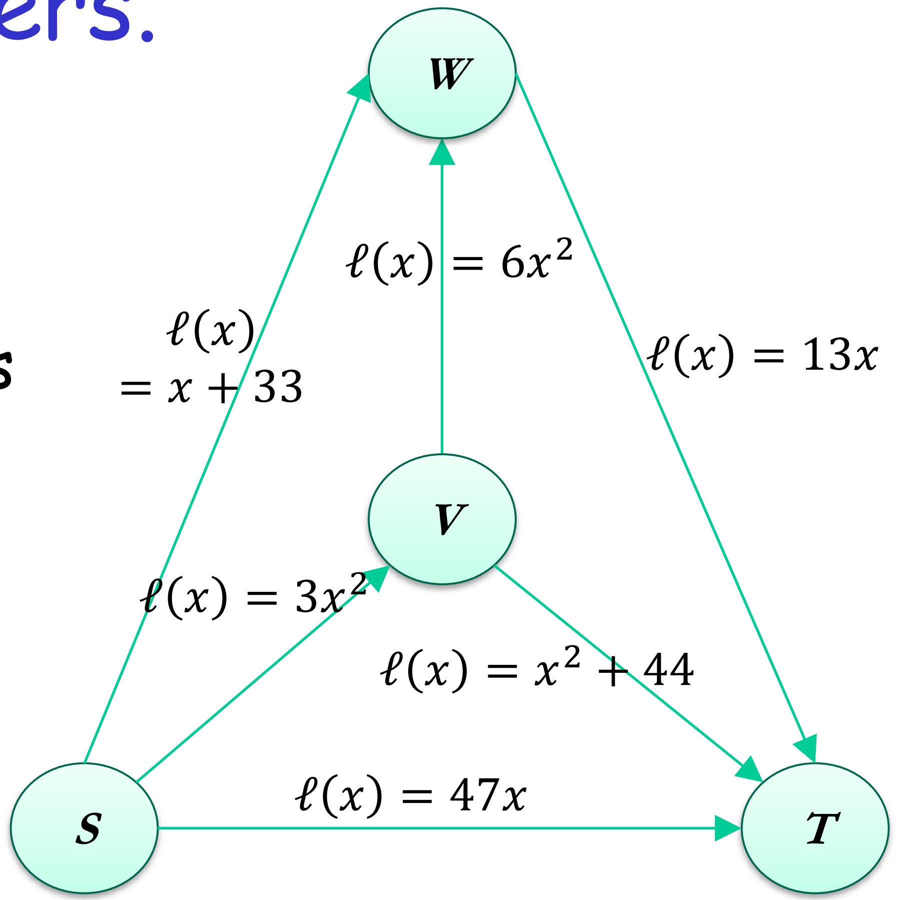


- $Total\ Cost = 2 \cdot 2 + 2 \cdot 2 + 1 \cdot 1 + 1 \cdot 1 = 10$

$$\Rightarrow PoA = \frac{10}{4} = 2.5$$

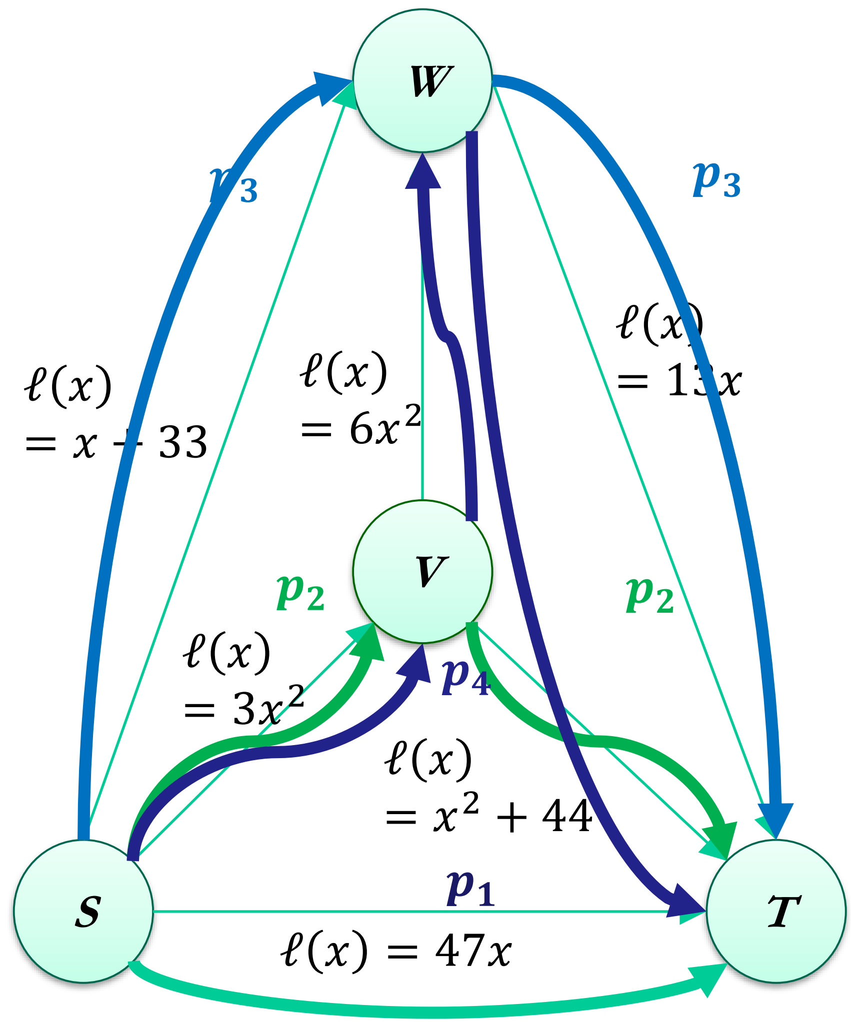
Unsplittable routing with weighted players.

- Let r_i denote the weight of player i.
- The loads and the payments are proportional to the weights.
  - Two players
  - $s_i = S, t_i = T$
  - $r_1 = 1, r_2 = 2$

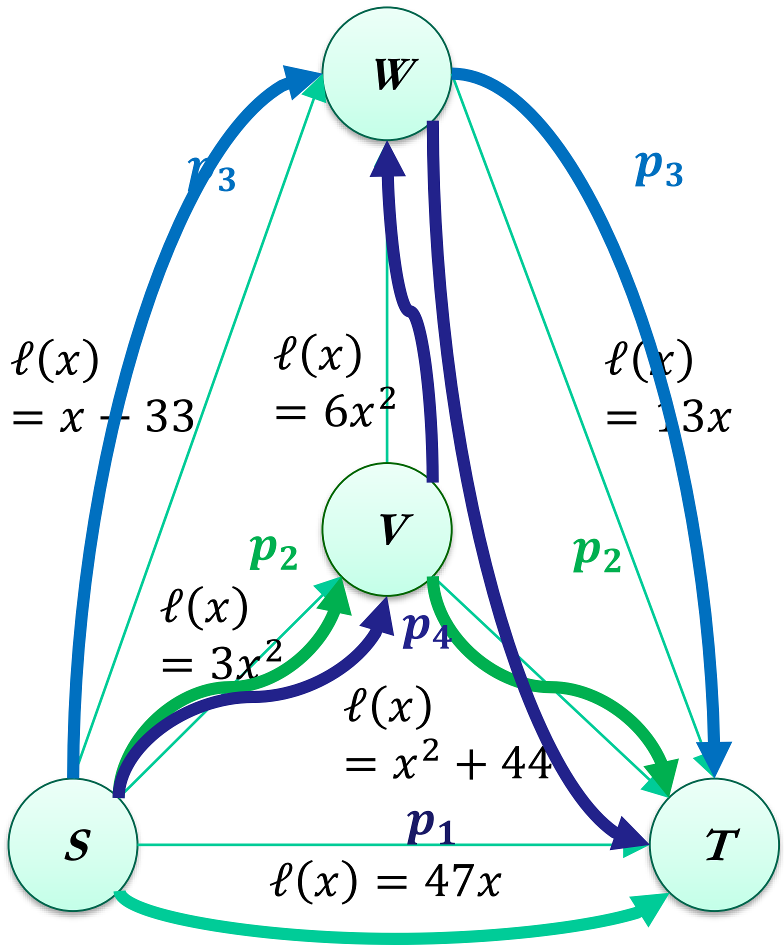


Unsplittable Routing with weighted players.

- There are four paths from S to T:
  - p_1: $S \to T$
  - p_2: $S \to V \to T$
  - p_3: $S \to W \to T$
  - p_4: $S \to V \to W \to T$



Weighted players. Cont.



Claim 1:

If Player 2 chooses either p_1 or p_2 then player 1 will choose p_4

Weighted players. Cont.

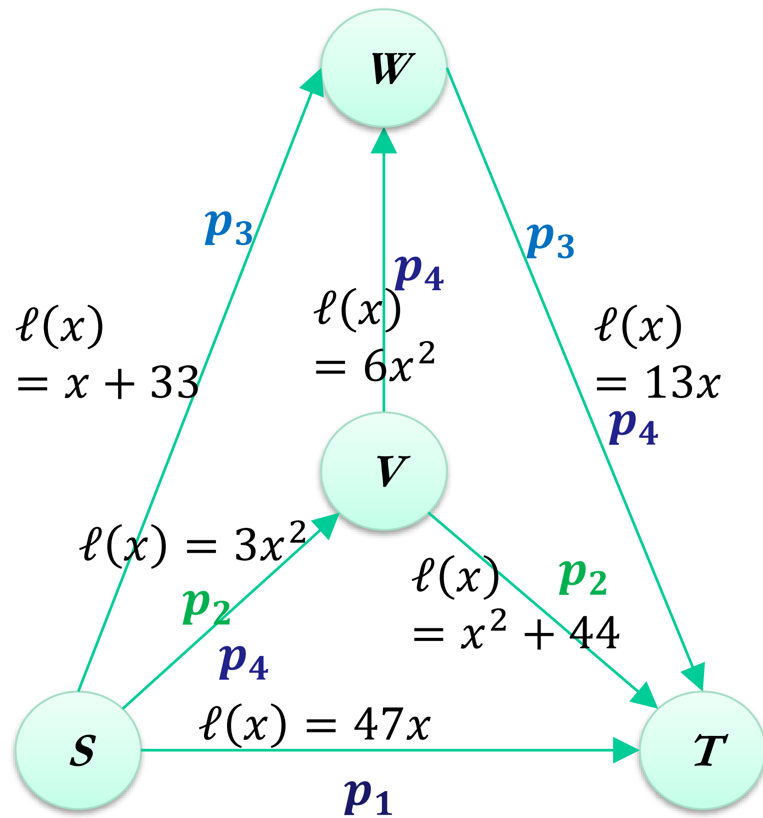
Proof of Claim 1:

Assuming player 2 chose path p_2, the cost of player 1 will be:

Path	Cost
p_1	47
p_2	$27 + 53 = 80$
p_3	$34 + 13 = 47$
p_4	$27 + 6 + 13 = 46$

Similarly, if player 2 choose path p_1, the cost of player 1 will be:

Path	Cost
p_1	141
p_2	$3 + 45 = 48$
p_3	$34 + 13 = 47$
p_4	$3 + 6 + 13 = 22$



Weighted players. Cont.

Similarly, it is possible to show the following claims:

1. Player 2 chooses either p_1 or p_2 → player 1 will choose p_4
2. Player 1 chooses path p_4 → player 2 will choose p_3
3. Player 2 chooses either p_3 or p_4 → player 1 will choose p_1
4. Player 1 chooses path p_1 → player 2 will choose p_2

Corollary: There is no pure NE in this game.

In general: A pure NE may not exist in weighted congestion games.