Computation and Improvement of Wardrop Equilibria with Unknown Demands

Max Klimm

1st part joint work with Philipp Warode

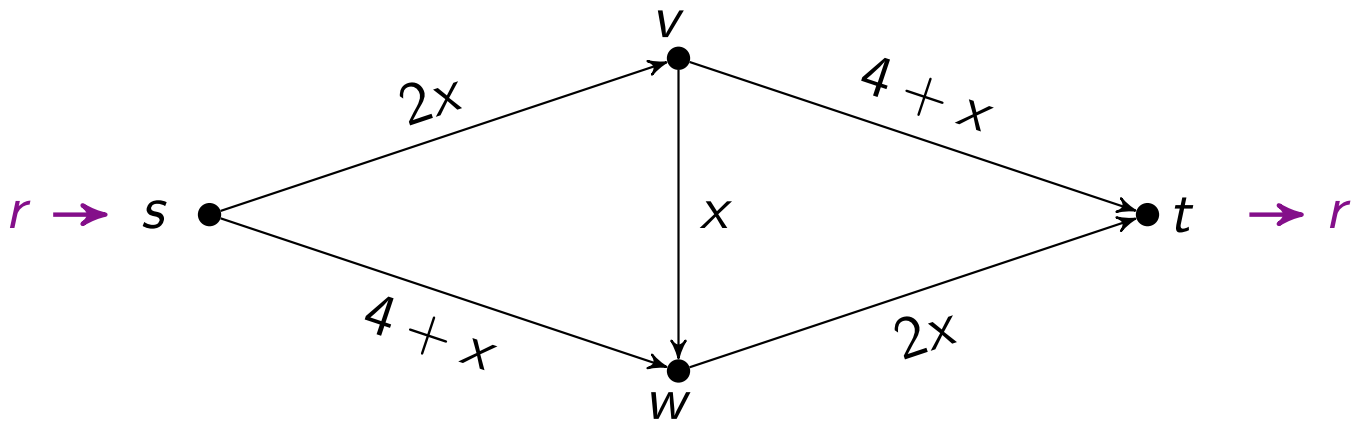
2nd part joint work with Marco Scarsini and Riccardo Colini-Baldeschi

UnRAVel guest talk RWTH Aachen (30.01.2019)

Problem Statement

Given:

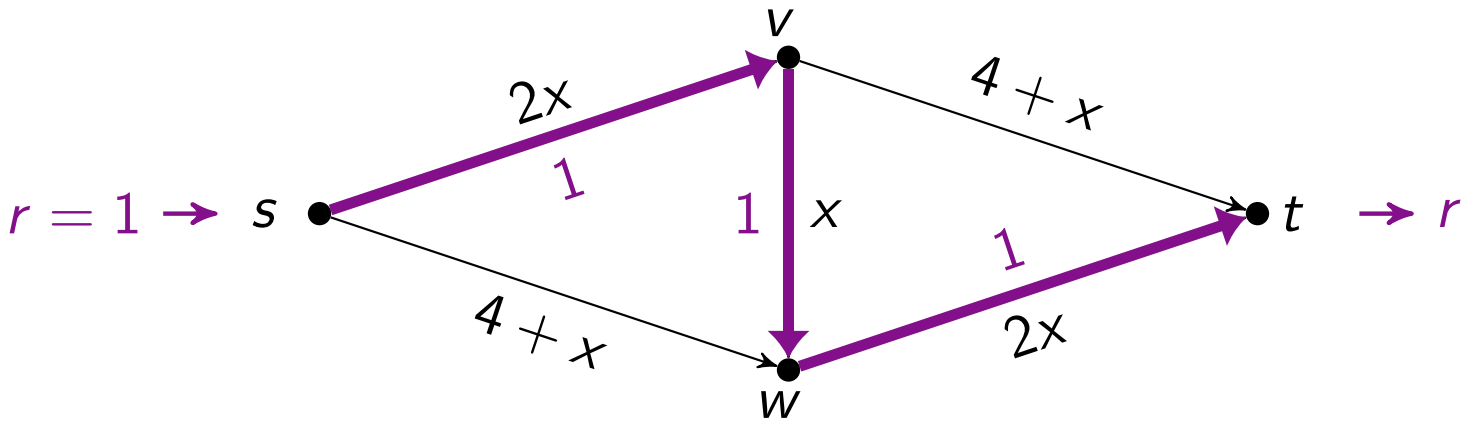
- graph G with vertex-edge incidence matrix $\mathbf{\Gamma}$
- increasing cost functions l_e
- source-/sink-vertices s, t



Problem Statement

Given:

- graph G with vertex-edge incidence matrix $\mathbf{\Gamma}$
- increasing cost functions l_e
- source-/sink-vertices s, t



Definition

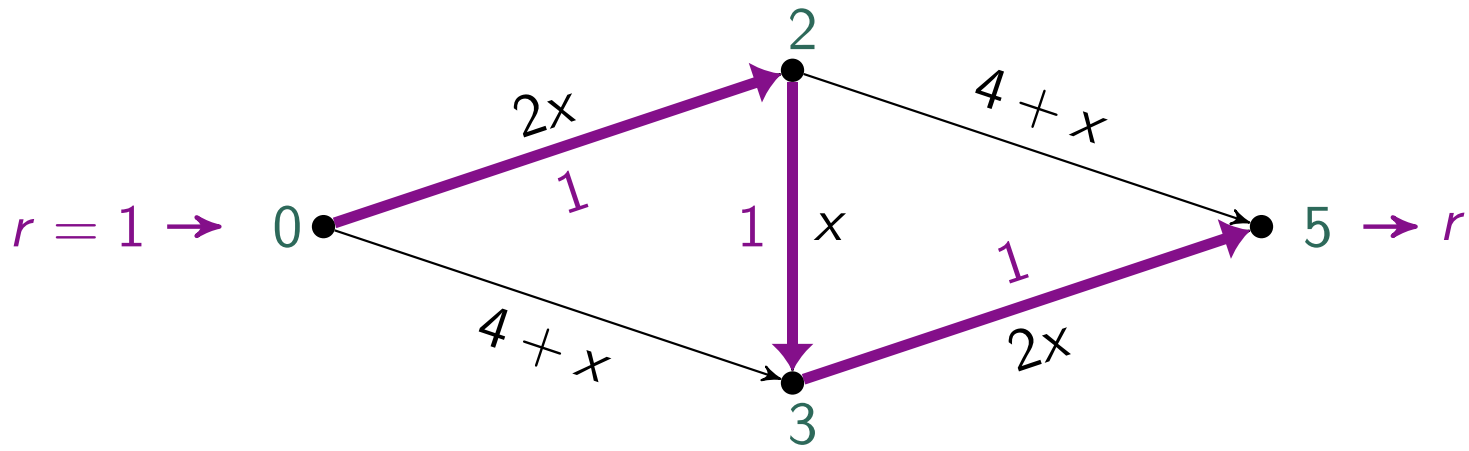
A feasible flow $\mathbf{x}$ is a Wardrop Equilibrium (WE) if

$$x_P > 0 \;\Rightarrow\; l_P(\mathbf{x}) \leq l_Q(\mathbf{x}) \qquad \forall P, Q \in \mathcal{P}.$$

Problem Statement

Given:

- graph G with vertex-edge incidence matrix $\mathbf{\Gamma}$
- increasing cost functions l_e
- source-/sink-vertices s, t



Lemma

A feasible flow $\mathbf{x}$ is a WE iff there is a vector $\boldsymbol{\pi} \in \mathbb{R}^V$ such that

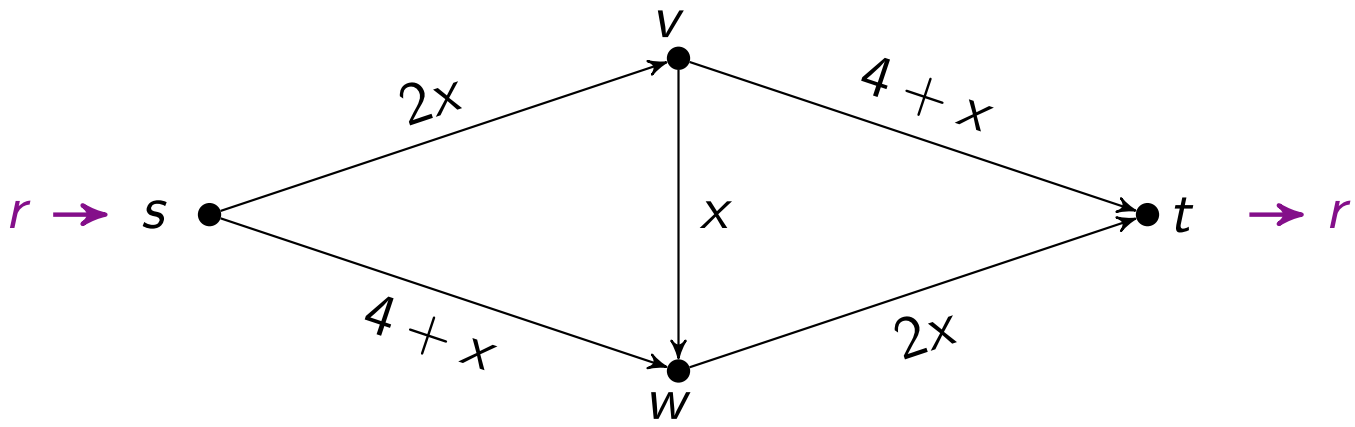
$$\pi_w - \pi_v = \boldsymbol{\gamma}_e^\top \boldsymbol{\pi} \leq l_e(x_e) \qquad \forall e \in E$$

$$\pi_w - \pi_v = \boldsymbol{\gamma}_e^\top \boldsymbol{\pi} = l_e(x_e) \qquad \forall e \in E : x_e > 0$$

Problem Statement

Given:

- graph G with vertex-edge incidence matrix $\mathbf{\Gamma}$
- increasing cost functions l_e
- source-/sink-vertices s, t



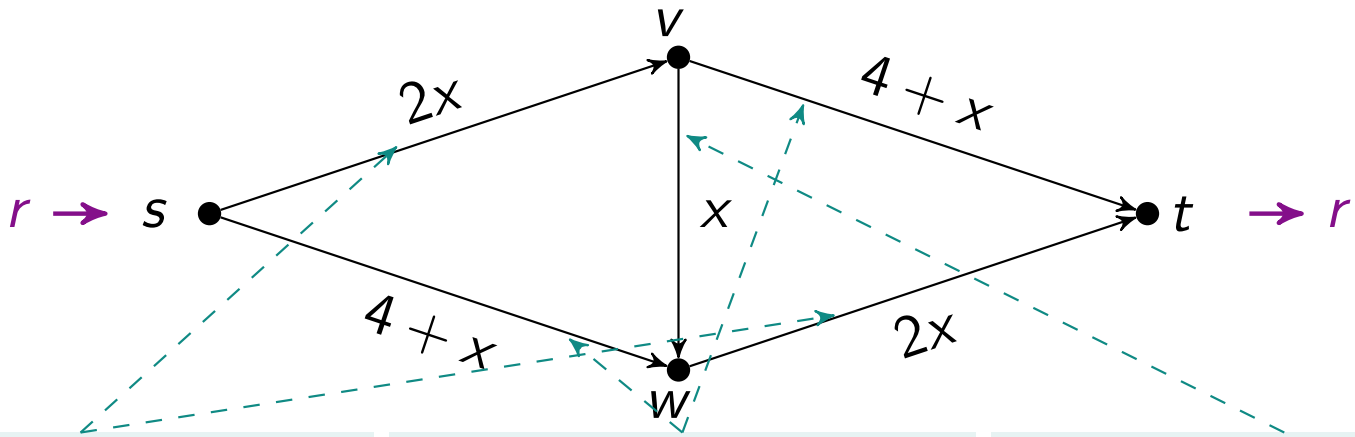
Goal:

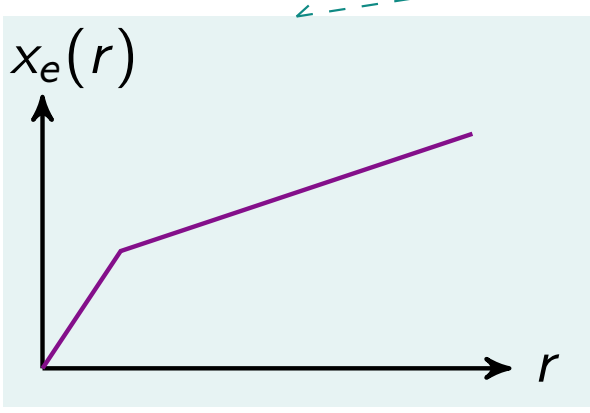
- functions $x_e : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ s.t.
  $\mathbf{x}(r) = (x_e(r))_{e \in E}$ is a WE with demand r for all $r \geq 0$

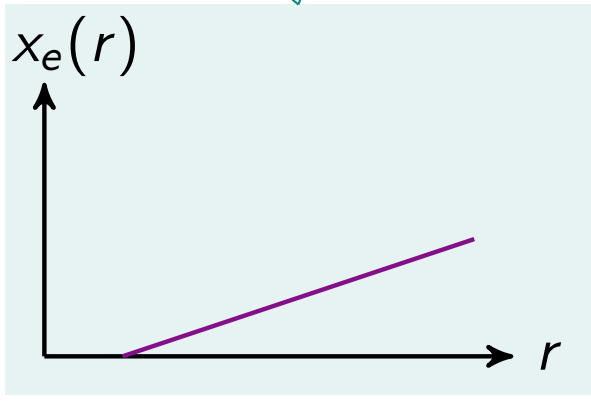
Problem Statement

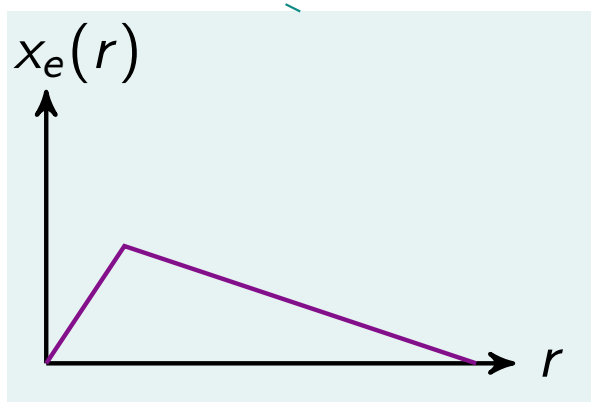
Given:

- graph G with vertex-edge incidence matrix $\mathbf{\Gamma}$
- increasing cost functions l_e
- source-/sink-vertices s, t





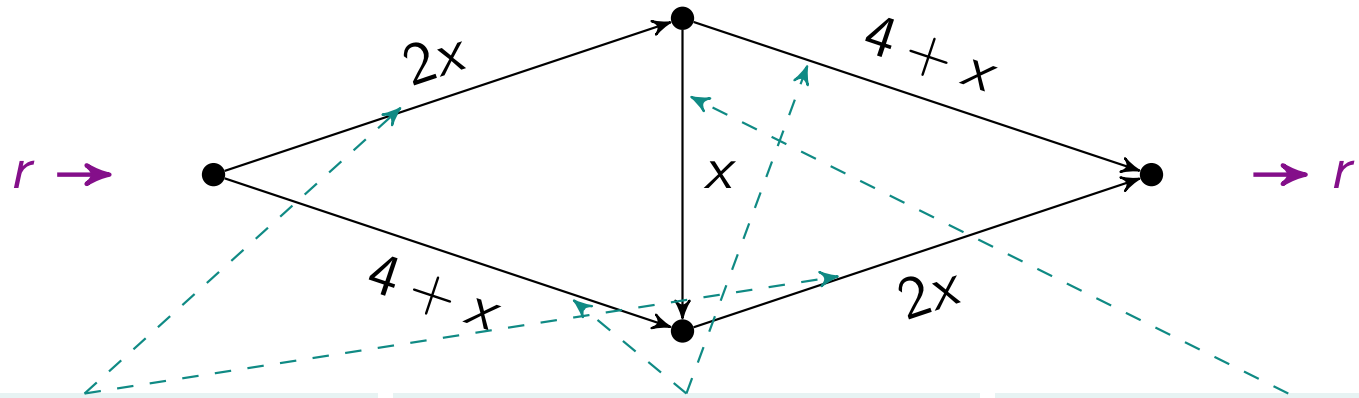


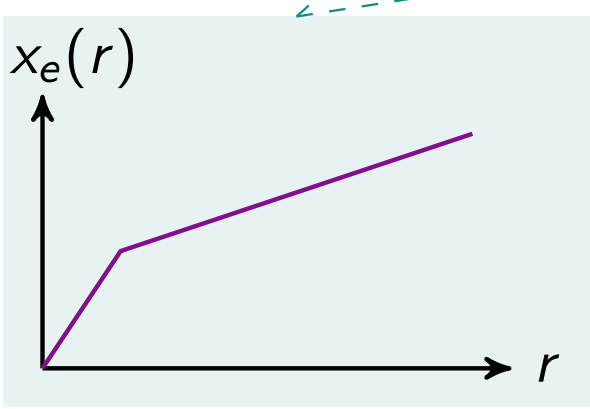


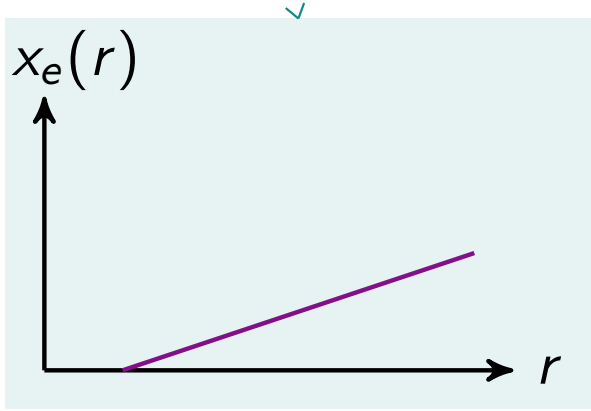
Problem Statement

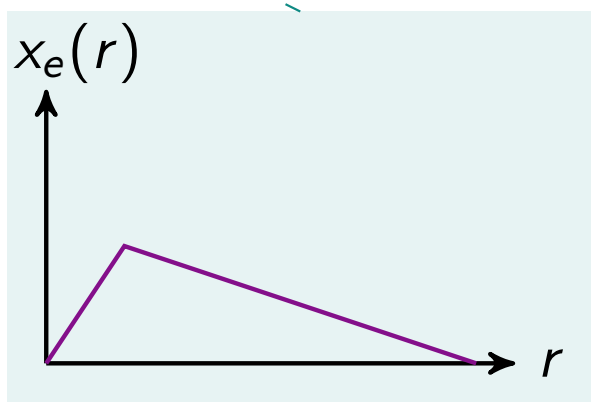
Given:

- graph G with vertex-edge incidence matrix $\mathbf{\Gamma}$
- increasing cost functions l_e piece-wise linear
- source-/sink-vertices s, t









Related Work

- Lemke's Algorithm for LCPs [Lemke 1965]
  - piece-wise linear extensions of Lemke's Algorithm [Kojima, Nishino, Hisakazu 1976]
  - all versions add an artificial variable which goes to zero in the course of the algorithm
  - intermediate solutions are not Wardrop equilibria

Related Work

- Lemke's Algorithm for LCPs [Lemke 1965]
  - piece-wise linear extensions of Lemke's Algorithm [Kojima, Nishino, Hisakazu 1976]
  - all versions add an artificial variable which goes to zero in the course of the algorithm
  - intermediate solutions are not Wardrop equilibria
- other homotopy methods in game theory [Herings, Peeters 2010]

Related Work

- Lemke's Algorithm for LCPs [Lemke 1965]
  - piece-wise linear extensions of Lemke's Algorithm [Kojima, Nishino, Hisakazu 1976]
  - all versions add an artificial variable which goes to zero in the course of the algorithm
  - intermediate solutions are not Wardrop equilibria
- other homotopy methods in game theory [Herings, Peeters 2010]
- sensitivity of Wardrop equilibria
  - continuity of path flows [Hall 1978]
  - generalizations to non-separable latencies [Dafermos, Nagurney 1984]
  - differentiability of flows [Patriksson 2004]
  - quantitative bounds [Englert, Franke, Olbrich 2008]

Related Work

- Lemke's Algorithm for LCPs [Lemke 1965]
  - piece-wise linear extensions of Lemke's Algorithm [Kojima, Nishino, Hisakazu 1976]
  - all versions add an artificial variable which goes to zero in the course of the algorithm
  - intermediate solutions are not Wardrop equilibria
- other homotopy methods in game theory [Herings, Peeters 2010]
- sensitivity of Wardrop equilibria
  - continuity of path flows [Hall 1978]
  - generalizations to non-separable latencies [Dafermos, Nagurney 1984]
  - differentiability of flows [Patriksson 2004]
  - quantitative bounds [Englert, Franke, Olbrich 2008]
- mechanism design for unknown demands
  - increase of cost functions [Christodoulou, Mehlhorn, Pyrga 2010]

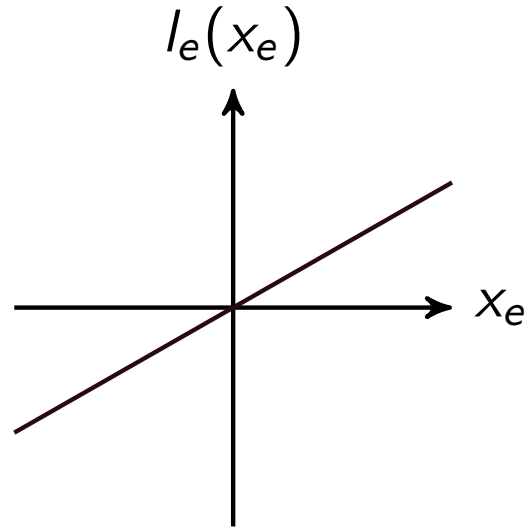
Computation via Electrical Flows

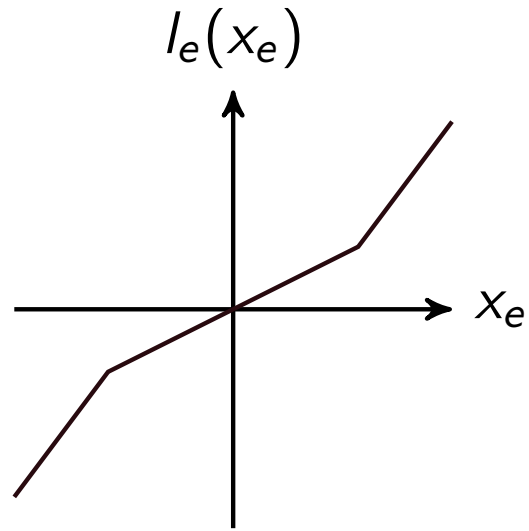
Electrical Flows

- in undirected networks with negative flow and $l(0) = 0$:

$$\pi_w - \pi_v = \boldsymbol{\gamma}_e^\top \boldsymbol{\pi} = l_e(x_e)$$

- equilibria correspond to electric flow
  - π - electrical potential (voltage)
  - x - electrical current
  - $l(x)$ - resistance
- homotopy method to compute electrical flows in networks with piecewise linear resistances [Katzenelson 1965]





The Framework

- repeat
  - start with a simple equilibrium potential
    (we use $\boldsymbol{\pi}^0 = 0$ for $r = 0$)
  - compute $\Delta\boldsymbol{\pi} = \boldsymbol{\pi}(r + \Delta r) - \boldsymbol{\pi}(r)$ for Δr small enough
    (piece-wise constant)
  - augment $\boldsymbol{\pi} = \boldsymbol{\pi} + \Delta\boldsymbol{\pi}$
- compute $\mathbf{x}$ from $\boldsymbol{\pi}$

The Framework

- repeat
  - start with a simple equilibrium potential
    (we use $\pi^0 = 0$ for $r = 0$)
  - compute $\Delta\pi = \pi(r + \Delta r) - \pi(r)$ for Δr small enough
    (piece-wise constant)
  - augment $\pi = \pi + \Delta\pi$
- compute $\mathbf{x}$ from π

Theorem [K. Warode, SODA 2019]

The functions $x(r)$ are k-wise linear and the above framework computes them in $\mathcal{O}(n^{2.4} + kn^2)$.

The Framework

- repeat
  - start with a simple equilibrium potential
    (we use $\pi^0 = 0$ for $r = 0$)
  - compute $\Delta\pi = \pi(r + \Delta r) - \pi(r)$ for Δr small enough
    (piece-wise constant)
  - augment $\pi = \pi + \Delta\pi$
- compute $\mathbf{x}$ from π

Theorem [K. Warode, SODA 2019]

The functions $x(r)$ are k-wise linear and the above framework computes them in $\mathcal{O}(n^{2.4} + kn^2)$.

- when approximating with arbitrary precision all exponents become close to 1

The Framework

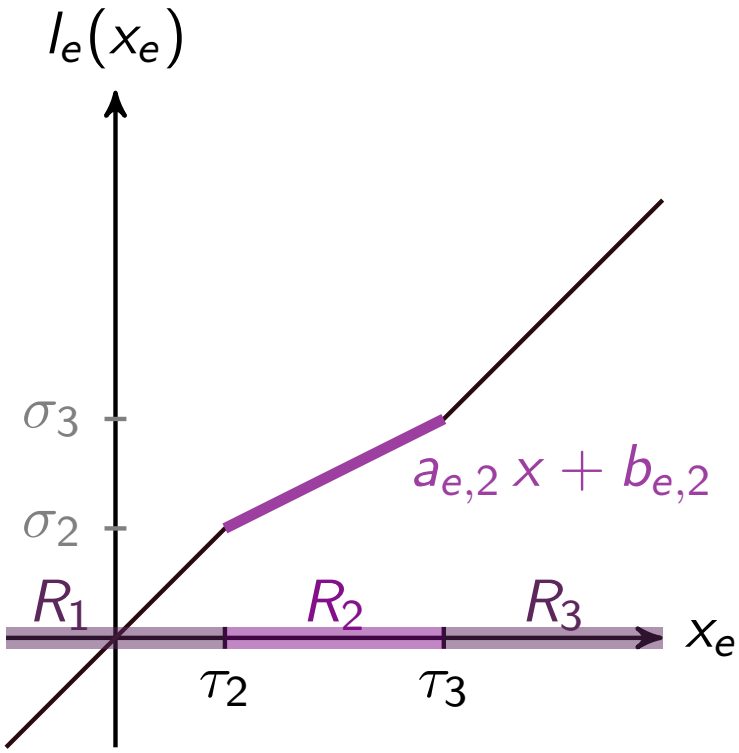
- repeat
  - start with a simple equilibrium potential
    (we use $\pi^0 = 0$ for $r = 0$)
  - compute $\Delta\pi = \pi(r + \Delta r) - \pi(r)$ for Δr small enough
    (piece-wise constant)
  - augment $\pi = \pi + \Delta\pi$
- compute $\mathbf{x}$ from π

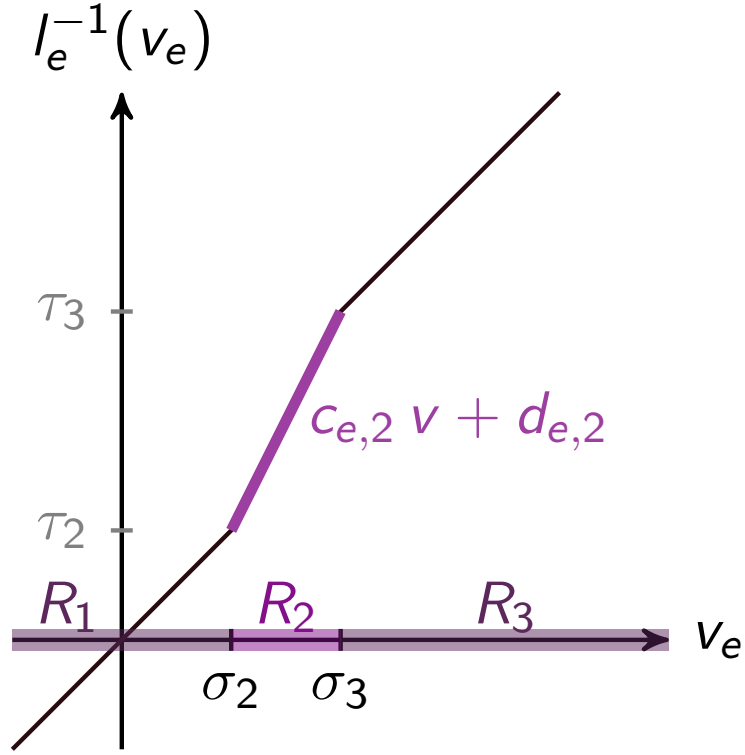
Theorem [K. Warode, SODA 2019]

The functions $x(r)$ are k-wise linear and the above framework computes them in $\mathcal{O}(n^{2.4} + kn^2)$.

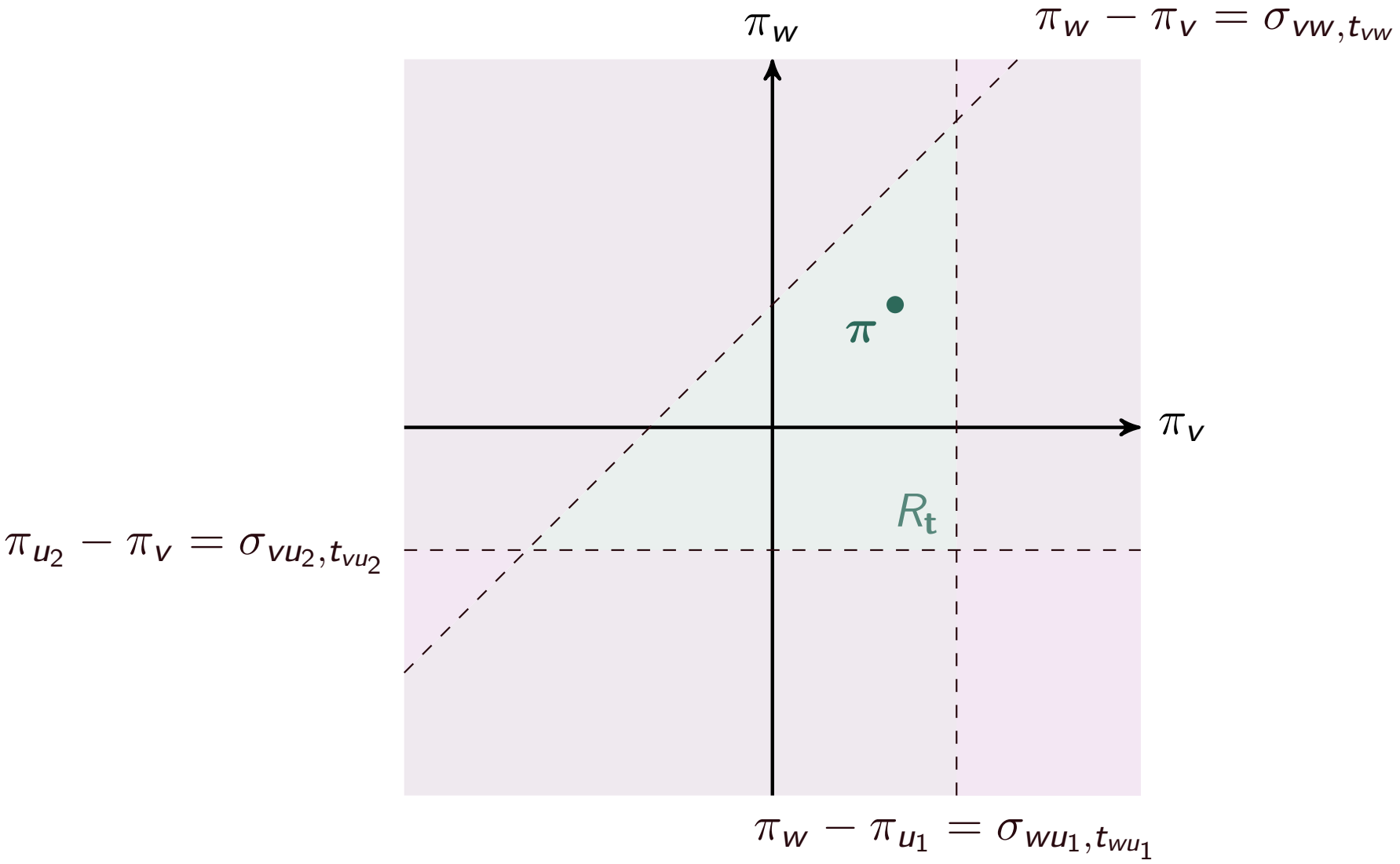
- when approximating with arbitrary precision all exponents become close to 1
- further results for degenerated points, directed edges, edge capacities, functions with constant parts, multi-commodity flows

Regions


$l_e(x_e)$
σ_3
σ_2
$a_{e,2}\,x + b_{e,2}$
R_1
R_2
R_3
x_e
τ_2
τ_3


$l_e^{-1}(v_e)$
τ_3
$c_{e,2}\,v + d_{e,2}$
τ_2
R_1
R_2
R_3
v_e
σ_2
σ_3

Potential Space



The Laplacian Matrix

Fix $\boldsymbol{\pi} \in R_{\mathbf{t}}$ for some region $R_{\mathbf{t}}$

The Laplacian Matrix

Fix $\boldsymbol{\pi} \in R_{\mathbf{t}}$ for some region $R_{\mathbf{t}}$

$$\textbf{flow} \quad x_e = l_e^{-1}(\pi_w - \pi_v) = l_e^{-1}(\boldsymbol{\gamma}_e^\top \boldsymbol{\pi}) = c_{e,t_e}\, \boldsymbol{\gamma}_e^\top \boldsymbol{\pi} + d_{e,t_e}$$

The Laplacian Matrix

Fix $\boldsymbol{\pi} \in R_{\mathbf{t}}$ for some region $R_{\mathbf{t}}$

$$\textbf{flow} \quad x_e = l_e^{-1}(\pi_w - \pi_v) = l_e^{-1}(\boldsymbol{\gamma}_e^\top \boldsymbol{\pi}) = c_{e,t_e}\, \boldsymbol{\gamma}_e^\top \boldsymbol{\pi} + d_{e,t_e}$$

$$\textbf{flow vector} \quad \mathbf{x} = \mathbf{C_t}\, \boldsymbol{\Gamma}^\top \boldsymbol{\pi} + \mathbf{d_t}$$

Fact: $\check{\mathbf{L}}_{\mathbf{t}}$ ($= \mathbf{L_t}$ without first row/column) is invertible

- one-to-one correspondence between (shortened) excess and potential

$$\check{\mathbf{y}} = \check{\mathbf{L}}_{\mathbf{t}}\, \check{\boldsymbol{\pi}} + \check{\mathbf{d}}_{\mathbf{t}}$$

$$\check{\boldsymbol{\pi}} = \check{\mathbf{L}}_{\mathbf{t}}^{-1}\, (\check{\mathbf{y}} - \check{\mathbf{d}}_{\mathbf{t}})$$

The Laplacian Matrix

Fix $\boldsymbol{\pi} \in R_{\mathbf{t}}$ for some region $R_{\mathbf{t}}$

flow $x_e = l_e^{-1}(\pi_w - \pi_v) = l_e^{-1}(\boldsymbol{\gamma}_e^\top \boldsymbol{\pi}) = c_{e,t_e}\, \boldsymbol{\gamma}_e^\top \boldsymbol{\pi} + d_{e,t_e}$

flow vector $\mathbf{x} = \mathbf{C_t}\, \boldsymbol{\Gamma}^\top \boldsymbol{\pi} + \mathbf{d_t}$

excess flow $y_v = \boldsymbol{\gamma}_v\, \mathbf{x} = \sum_{(u,v)\in E} x_e - \sum_{(v,u)\in E} x_e$

Fact: $\check{\mathbf{L}}_{\mathbf{t}}$ ($= \mathbf{L_t}$ without first row/column) is invertible

- one-to-one correspondence between (shortened) excess and potential

$$\check{\mathbf{y}} = \check{\mathbf{L}}_{\mathbf{t}}\, \check{\boldsymbol{\pi}} + \check{\mathbf{d}}_{\mathbf{t}}$$

$$\check{\boldsymbol{\pi}} = \check{\mathbf{L}}_{\mathbf{t}}^{-1}\, (\check{\mathbf{y}} - \check{\mathbf{d}}_{\mathbf{t}})$$

The Laplacian Matrix

Fix $\boldsymbol{\pi} \in R_{\mathbf{t}}$ for some region $R_{\mathbf{t}}$

flow $x_e = l_e^{-1}(\pi_w - \pi_v) = l_e^{-1}(\boldsymbol{\gamma}_e^\top \boldsymbol{\pi}) = c_{e,t_e}\, \boldsymbol{\gamma}_e^\top \boldsymbol{\pi} + d_{e,t_e}$

flow vector $\mathbf{x} = \mathbf{C_t}\, \boldsymbol{\Gamma}^\top \boldsymbol{\pi} + \mathbf{d_t}$

excess flow $y_v = \boldsymbol{\gamma}_v\, \mathbf{x} = \sum_{(u,v)\in E} x_e - \sum_{(v,u)\in E} x_e$

excess flow vector $\mathbf{y} = \boldsymbol{\Gamma}\, \mathbf{C_t}\, \boldsymbol{\Gamma}^\top \boldsymbol{\pi} + \boldsymbol{\Gamma}\, \mathbf{d_t} = \mathbf{L_t}\, \boldsymbol{\pi} + \tilde{\mathbf{d}}_\mathbf{t}$ with $\tilde{\mathbf{d}}_\mathbf{t} = \boldsymbol{\Gamma} \mathbf{d_t}$

Fact: $\check{\mathbf{L}}_\mathbf{t}$ ($= \mathbf{L_t}$ without first row/column) is invertible

- one-to-one correspondence between (shortened) excess and potential

$$\check{\mathbf{y}} = \check{\mathbf{L}}_\mathbf{t}\, \check{\boldsymbol{\pi}} + \check{\mathbf{d}}_\mathbf{t}$$

$$\check{\boldsymbol{\pi}} = \check{\mathbf{L}}_\mathbf{t}^{-1}\, (\check{\mathbf{y}} - \check{\mathbf{d}}_\mathbf{t})$$

The Laplacian Matrix

Fix $\boldsymbol{\pi} \in R_{\mathbf{t}}$ for some region $R_{\mathbf{t}}$

flow $x_e = l_e^{-1}(\pi_w - \pi_v) = l_e^{-1}(\boldsymbol{\gamma}_e^\top \boldsymbol{\pi}) = c_{e,t_e}\, \boldsymbol{\gamma}_e^\top \boldsymbol{\pi} + d_{e,t_e}$

flow vector $\mathbf{x} = \mathbf{C_t}\, \boldsymbol{\Gamma}^\top \boldsymbol{\pi} + \mathbf{d_t}$

excess flow $y_v = \boldsymbol{\gamma}_v\, \mathbf{x} = \sum_{(u,v)\in E} x_e - \sum_{(v,u)\in E} x_e$

excess flow vector $\mathbf{y} = \boldsymbol{\Gamma}\, \mathbf{C_t}\, \boldsymbol{\Gamma}^\top \boldsymbol{\pi} + \boldsymbol{\Gamma}\, \mathbf{d_t} = \mathbf{L_t}\, \boldsymbol{\pi} + \tilde{\mathbf{d}}_\mathbf{t}$ with $\tilde{\mathbf{d}}_\mathbf{t} = \boldsymbol{\Gamma} \mathbf{d_t}$

Laplacian Matrix $\mathbf{L_t} = \boldsymbol{\Gamma}\, \mathbf{C_t}\, \boldsymbol{\Gamma}^\top$ (of region $R_{\mathbf{t}}$)

Fact: $\check{\mathbf{L}}_\mathbf{t}$ ($= \mathbf{L_t}$ without first row/column) is invertible

- one-to-one correspondence between (shortened) excess and potential

$$\check{\mathbf{y}} = \check{\mathbf{L}}_\mathbf{t}\, \check{\boldsymbol{\pi}} + \check{\mathbf{d}}_\mathbf{t}$$

$$\check{\boldsymbol{\pi}} = \check{\mathbf{L}}_\mathbf{t}^{-1}\, (\check{\mathbf{y}} - \check{\mathbf{d}}_\mathbf{t})$$

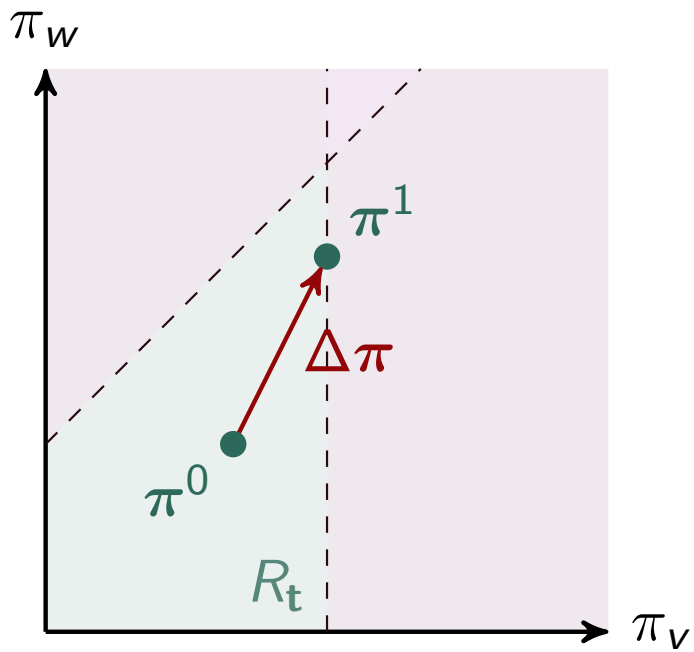
Direction Vectors

- increasing demand by $\Delta r = 1$ changes excess by

$$\Delta \mathbf{y} = \mathbb{1}_t - \mathbb{1}_s$$

and (shortened) potential by

$$\Delta \check{\boldsymbol{\pi}} = \check{\mathbf{L}}_{\mathbf{t}}^{-1} \, \Delta \check{\mathbf{y}} = \check{\mathbf{L}}_{\mathbf{t}}^{-1} (\check{\mathbb{1}}_t - \check{\mathbb{1}}_s)$$



- solution curve moves along $\Delta\pi$ in the potential space until boundary is hit

Boundary Crossing

- first boundary hit is

$$\arg\min_{e\in E} \epsilon(e) := \arg\min_{e\in E} \begin{cases} \frac{\sigma_{\mathbf{t}_e+1} - \boldsymbol{\gamma}_e^\top \boldsymbol{\pi}}{\boldsymbol{\gamma}_e^\top \Delta\boldsymbol{\pi}} & \text{if } \boldsymbol{\gamma}_e^\top \Delta\boldsymbol{\pi} > 0 \\ \frac{\sigma_{\mathbf{t}_e} - \boldsymbol{\gamma}_e^\top \boldsymbol{\pi}}{\boldsymbol{\gamma}_e^\top \Delta\boldsymbol{\pi}} & \text{if } \boldsymbol{\gamma}_e^\top \Delta\boldsymbol{\pi} < 0 \\ \infty & \text{otherwise} \end{cases}$$

Boundary Crossing

- first boundary hit is

$$\arg\min_{e\in E} \epsilon(e) := \arg\min_{e\in E} \begin{cases} \frac{\sigma_{\mathbf{t}_e+1} - \boldsymbol{\gamma}_e^\top \boldsymbol{\pi}}{\boldsymbol{\gamma}_e^\top \Delta\boldsymbol{\pi}} & \text{if } \boldsymbol{\gamma}_e^\top \Delta\boldsymbol{\pi} > 0 \\ \frac{\sigma_{\mathbf{t}_e} - \boldsymbol{\gamma}_e^\top \boldsymbol{\pi}}{\boldsymbol{\gamma}_e^\top \Delta\boldsymbol{\pi}} & \text{if } \boldsymbol{\gamma}_e^\top \Delta\boldsymbol{\pi} < 0 \\ \infty & \text{otherwise} \end{cases}$$

- new Laplace Matrix is

$$\begin{aligned} \mathbf{L}_{\mathbf{t}^2} = \boldsymbol{\Gamma}\, \mathbf{C}_{\mathbf{t}^2}\, \boldsymbol{\Gamma}^\top &= \boldsymbol{\Gamma} \left(\mathbf{C}_{\mathbf{t}^1} + \operatorname{diag}(0, \ldots, 0, \Delta c_e, 0, \ldots, 0) \right) \boldsymbol{\Gamma}^\top \\ &= \mathbf{L}_{\mathbf{t}^1} + \Delta c_e\, \boldsymbol{\gamma}_e\, \boldsymbol{\gamma}_e^\top \end{aligned}$$

Boundary Crossing

- first boundary hit is

$$\arg\min_{e\in E} \epsilon(e) := \arg\min_{e\in E} \begin{cases} \frac{\sigma_{\mathbf{t}_e+1} - \boldsymbol{\gamma}_e^\top \boldsymbol{\pi}}{\boldsymbol{\gamma}_e^\top \Delta\boldsymbol{\pi}} & \text{if } \boldsymbol{\gamma}_e^\top \Delta\boldsymbol{\pi} > 0 \\ \frac{\sigma_{\mathbf{t}_e} - \boldsymbol{\gamma}_e^\top \boldsymbol{\pi}}{\boldsymbol{\gamma}_e^\top \Delta\boldsymbol{\pi}} & \text{if } \boldsymbol{\gamma}_e^\top \Delta\boldsymbol{\pi} < 0 \\ \infty & \text{otherwise} \end{cases}$$

- new Laplace Matrix is

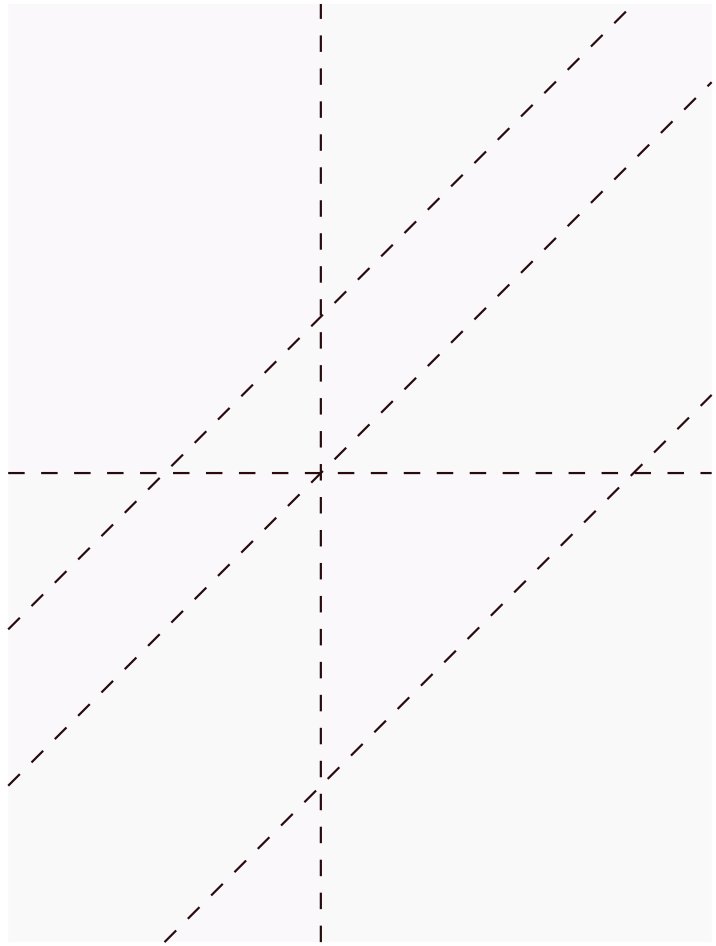
$$\begin{aligned} \mathbf{L}_{\mathbf{t}^2} = \boldsymbol{\Gamma}\, \mathbf{C}_{\mathbf{t}^2}\, \boldsymbol{\Gamma}^\top &= \boldsymbol{\Gamma}\left(\mathbf{C}_{\mathbf{t}^1} + \operatorname{diag}(0,\dots,0,\Delta c_e,0,\dots,0)\right)\boldsymbol{\Gamma}^\top \\ &= \mathbf{L}_{\mathbf{t}^1} + \Delta c_e\, \boldsymbol{\gamma}_e\, \boldsymbol{\gamma}_e^\top \end{aligned}$$

Theorem (Fujisawa, Kuh '72)

$$\check{\mathbf{L}}_{\mathbf{t}^2}^{-1} = \check{\mathbf{L}}_{\mathbf{t}^1}^{-1}\left(\mathbf{I}_{n-1} - \frac{\Delta c_e}{1 + \Delta c_e \check{\boldsymbol{\gamma}}_e^\top \check{\mathbf{L}}_{\mathbf{t}^1}^{-1} \check{\boldsymbol{\gamma}}_e} \check{\boldsymbol{\gamma}}_e \check{\boldsymbol{\gamma}}_e^\top \check{\mathbf{L}}_{\mathbf{t}^1}^{-1}\right)$$

Katzenelson's Algorithm

Find initial potential $\boldsymbol{\pi}^0 \in R_{\mathbf{t}^0}$;
Compute $\check{\mathbf{L}}^{-1}_{\mathbf{t}^0}$;
while true do
  Compute direction $\Delta\check{\boldsymbol{\pi}}^i$;
  $\epsilon = \min_{e \in E}$ distance to boundary;
  if $\epsilon = \infty$ **then**
    return;
  end
  $\check{\boldsymbol{\pi}}^{i+1} = \check{\boldsymbol{\pi}}^i + \epsilon\Delta\check{\boldsymbol{\pi}}^i$;
  Compute $\check{\mathbf{L}}^{-1}_{\mathbf{t}^{i+1}}$ for $R_{\mathbf{t}^{i+1}}$;
end



Katzenelson's Algorithm

Find initial potential $\boldsymbol{\pi}^0 \in R_{\mathbf{t}^0}$;

Compute $\check{\mathbf{L}}_{\mathbf{t}}^{-1}$;

while true do

  Compute direction $\Delta\check{\boldsymbol{\pi}}^i$;

  $\epsilon = \min_{e \in E}$ distance to boundary;

  if $\epsilon = \infty$ **then**

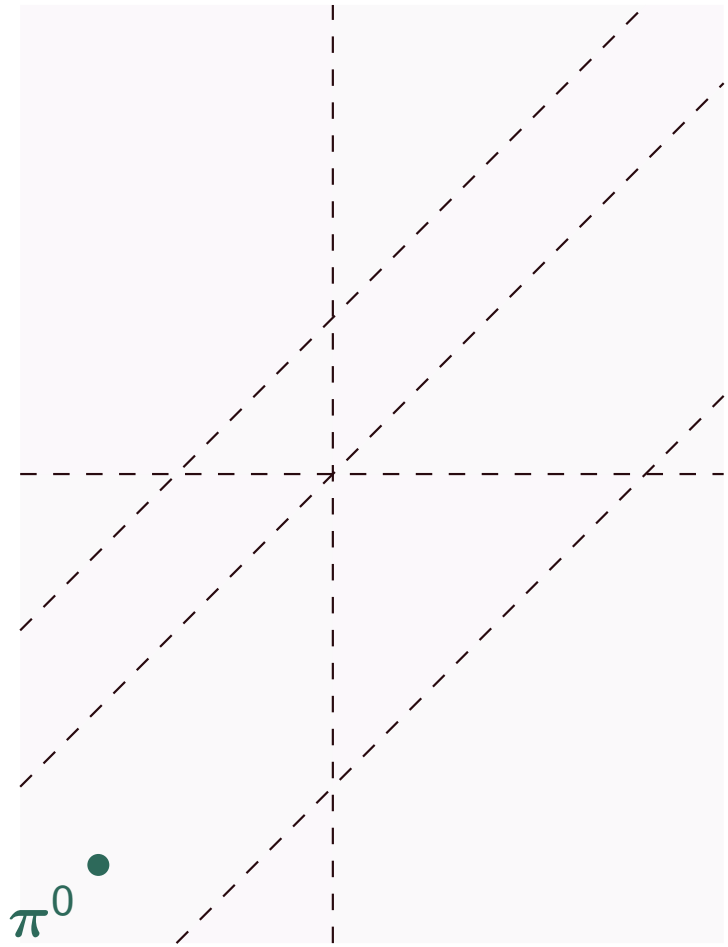
    return;

  end

  $\check{\boldsymbol{\pi}}^{i+1} = \check{\boldsymbol{\pi}}^i + \epsilon \Delta\check{\boldsymbol{\pi}}^i$;

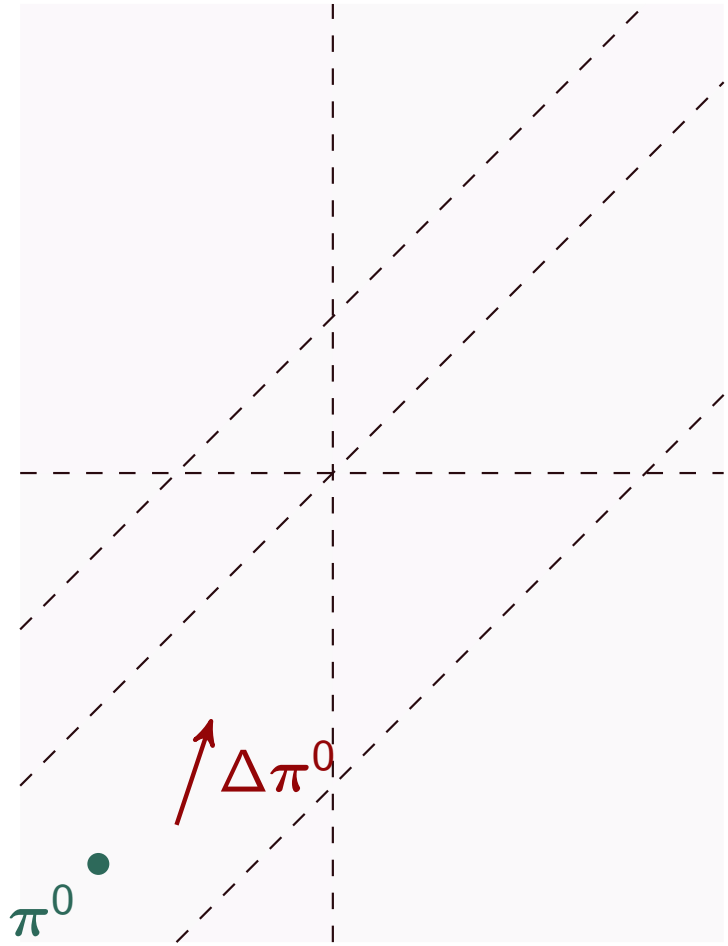
  Compute $\check{\mathbf{L}}_{\mathbf{t}^{i+1}}^{-1}$ for $R_{\mathbf{t}^{i+1}}$;

end



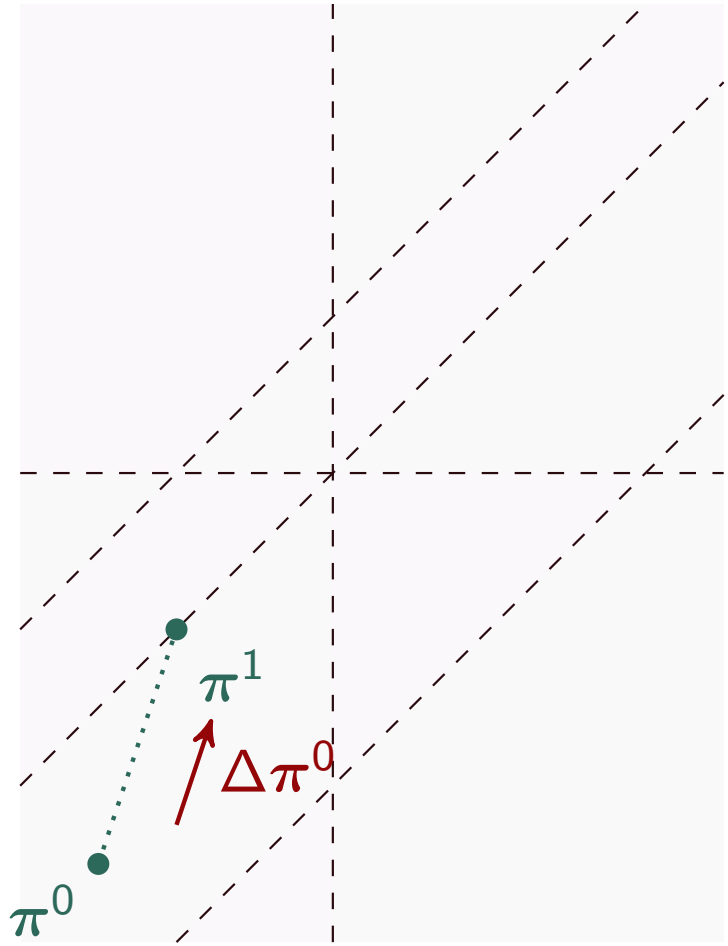
Katzenelson's Algorithm

Find initial potential $\boldsymbol{\pi}^0 \in R_{\mathbf{t}^0}$;
Compute $\check{\mathbf{L}}^{-1}_{\mathbf{t}^0}$;
while true do
  Compute direction $\Delta\check{\boldsymbol{\pi}}^i$;
  $\epsilon = \min_{e \in E}$ distance to boundary;
  if $\epsilon = \infty$ **then**
    return;
  end
  $\check{\boldsymbol{\pi}}^{i+1} = \check{\boldsymbol{\pi}}^i + \epsilon \Delta\check{\boldsymbol{\pi}}^i$;
  Compute $\check{\mathbf{L}}^{-1}_{\mathbf{t}^{i+1}}$ for $R_{\mathbf{t}^{i+1}}$;
end



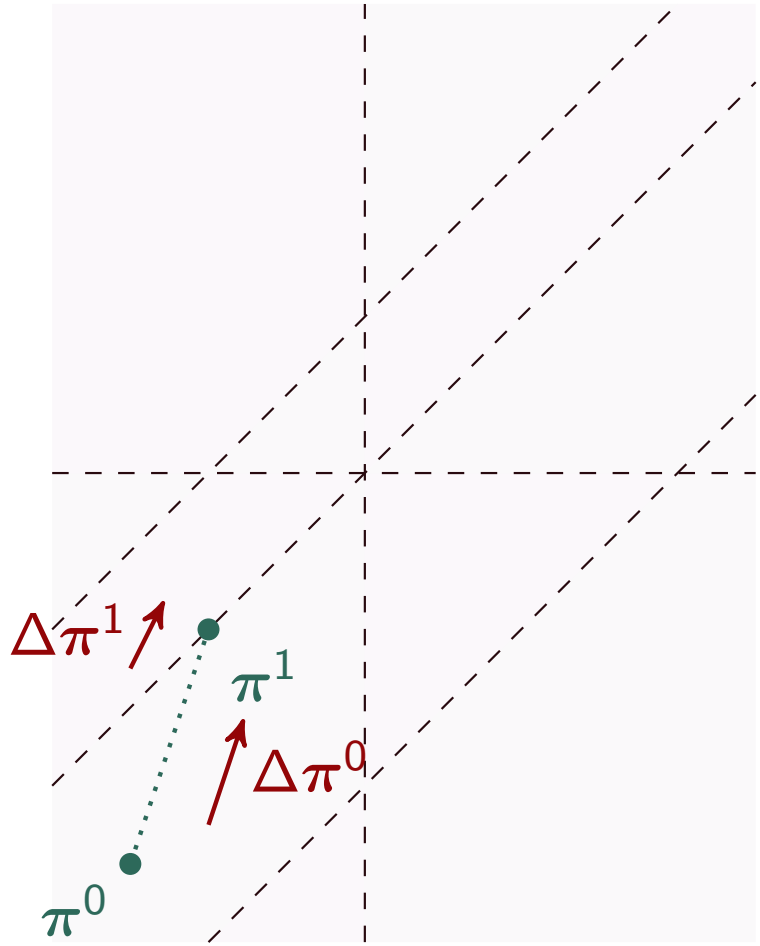
Katzenelson's Algorithm

Find initial potential $\boldsymbol{\pi}^0 \in R_{\mathbf{t}^0}$;
Compute $\check{\mathbf{L}}^{-1}_{\mathbf{t}^0}$;
while true do
  Compute direction $\Delta\check{\boldsymbol{\pi}}^i$;
  $\epsilon = \min_{e \in E}$ distance to boundary;
  if $\epsilon = \infty$ **then**
    return;
  end
  $\check{\boldsymbol{\pi}}^{i+1} = \check{\boldsymbol{\pi}}^i + \epsilon\Delta\check{\boldsymbol{\pi}}^i$;
  Compute $\check{\mathbf{L}}^{-1}_{\mathbf{t}^{i+1}}$ for $R_{\mathbf{t}^{i+1}}$;
end



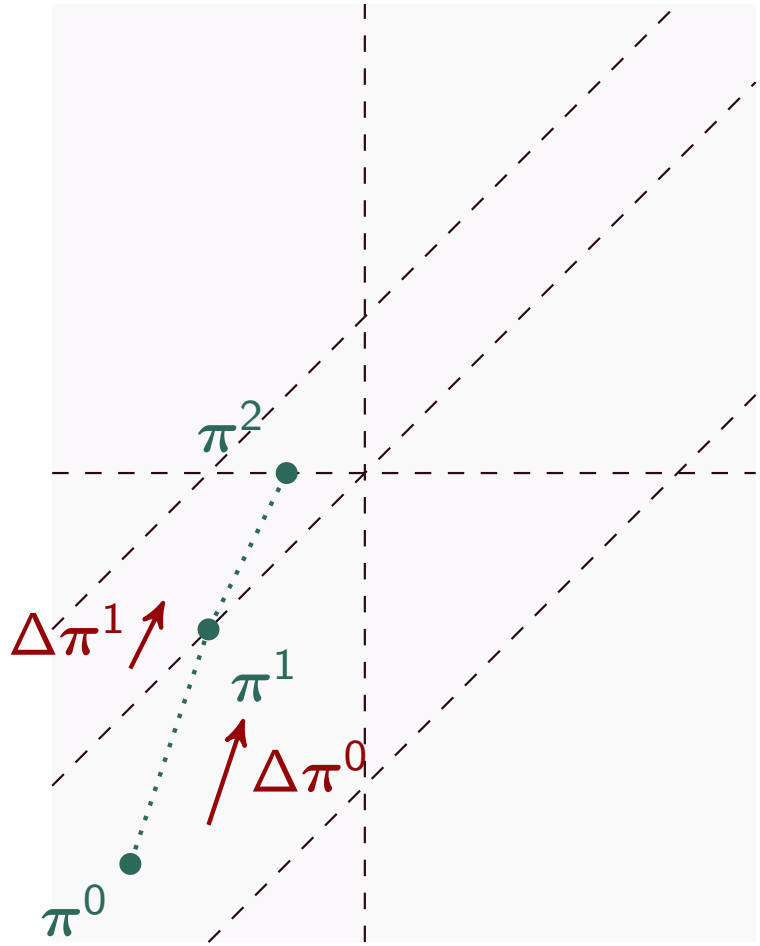
Katzenelson's Algorithm

Find initial potential $\boldsymbol{\pi}^0 \in R_{\mathbf{t}^0}$;
Compute $\check{\mathbf{L}}^{-1}_{\mathbf{t}^0}$;
while true do
  Compute direction $\Delta\check{\boldsymbol{\pi}}^i$;
  $\epsilon = \min_{e \in E}$ distance to boundary;
  if $\epsilon = \infty$ **then**
    return;
  end
  $\check{\boldsymbol{\pi}}^{i+1} = \check{\boldsymbol{\pi}}^i + \epsilon \Delta\check{\boldsymbol{\pi}}^i$;
  Compute $\check{\mathbf{L}}^{-1}_{\mathbf{t}^{i+1}}$ for $R_{\mathbf{t}^{i+1}}$;
end



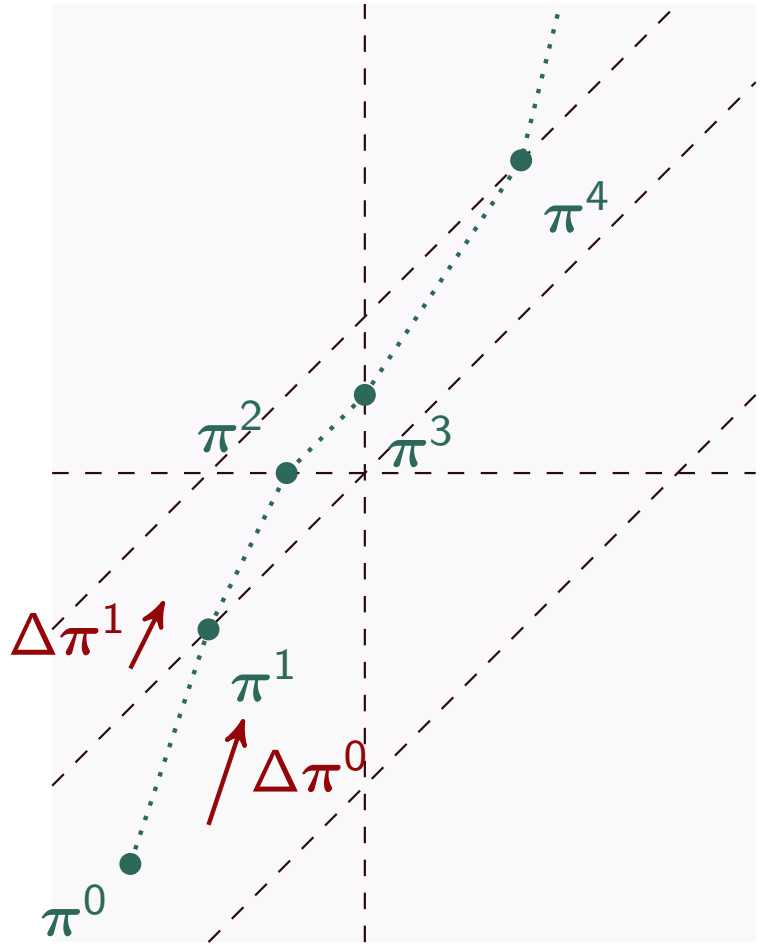
Katzenelson's Algorithm

Find initial potential $\boldsymbol{\pi}^0 \in R_{\mathbf{t}^0}$;
Compute $\check{\mathbf{L}}^{-1}_{\mathbf{t}^0}$;
while true do
  Compute direction $\Delta\check{\boldsymbol{\pi}}^i$;
  $\epsilon = \min_{e \in E}$ distance to boundary;
  if $\epsilon = \infty$ **then**
    return;
  end
  $\check{\boldsymbol{\pi}}^{i+1} = \check{\boldsymbol{\pi}}^i + \epsilon \Delta\check{\boldsymbol{\pi}}^i$;
  Compute $\check{\mathbf{L}}^{-1}_{\mathbf{t}^{i+1}}$ for $R_{\mathbf{t}^{i+1}}$;
end



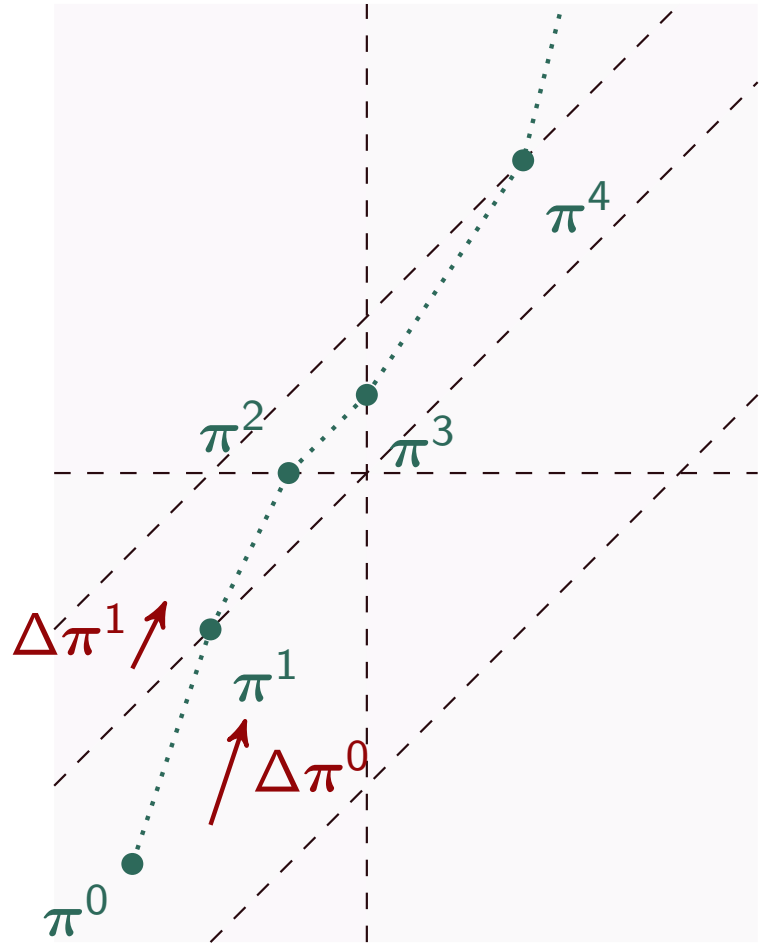
Katzenelson's Algorithm

Find initial potential $\boldsymbol{\pi}^0 \in R_{\mathbf{t}^0}$;
Compute $\check{\mathbf{L}}^{-1}_{\mathbf{t}^0}$;
while true do
  Compute direction $\Delta\check{\boldsymbol{\pi}}^i$;
  $\epsilon = \min_{e \in E}$ distance to boundary;
  if $\epsilon = \infty$ **then**
    return;
  end
  $\check{\boldsymbol{\pi}}^{i+1} = \check{\boldsymbol{\pi}}^i + \epsilon \Delta\check{\boldsymbol{\pi}}^i$;
  Compute $\check{\mathbf{L}}^{-1}_{\mathbf{t}^{i+1}}$ for $R_{\mathbf{t}^{i+1}}$;
end



Katzenelson's Algorithm

Find initial potential $\boldsymbol{\pi}^0 \in R_{\mathbf{t}^0}$;
Compute $\check{\mathbf{L}}^{-1}_{\mathbf{t}^0}$;
while true do
  Compute direction $\Delta\check{\boldsymbol{\pi}}^i$;
  $\epsilon = \min_{e \in E}$ distance to boundary;
  if $\epsilon = \infty$ **then**
    return;
  end
  $\check{\boldsymbol{\pi}}^{i+1} = \check{\boldsymbol{\pi}}^i + \epsilon \Delta\check{\boldsymbol{\pi}}^i$;
  Compute $\check{\mathbf{L}}^{-1}_{\mathbf{t}^{i+1}}$ for $R_{\mathbf{t}^{i+1}}$;
end

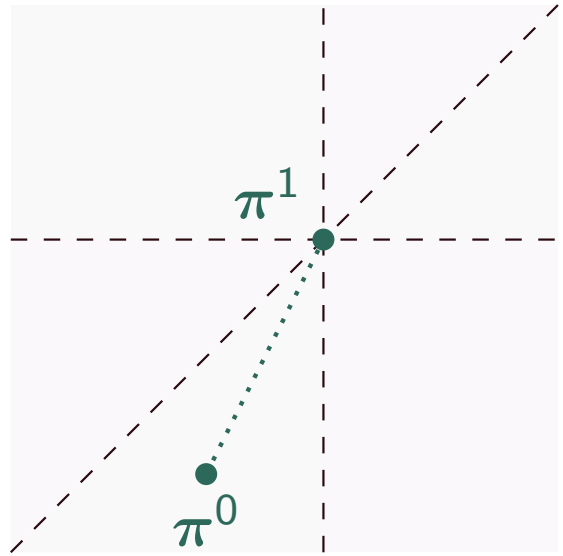


Solution: Piecewise linear function $\boldsymbol{\pi}(r) = \boldsymbol{\pi}^i + (r - r^i)\,\Delta\boldsymbol{\pi}^i$
$\rightarrow$ Piecewise linear flow functions $\mathbf{x}(r) = \mathbf{x}^i + (r - r^i)\,\Delta\mathbf{x}^i$
where $\mathbf{x}^i = \mathbf{C}_{\mathbf{t}^i}\,\boldsymbol{\Gamma}^\top\,\boldsymbol{\pi}^i + \mathbf{d}_{\mathbf{t}^i}$ and $\Delta\mathbf{x}^i = \mathbf{C}_{\mathbf{t}^i}\,\boldsymbol{\Gamma}^\top\,\Delta\boldsymbol{\pi}^i$

Degeneracy

Degeneracy

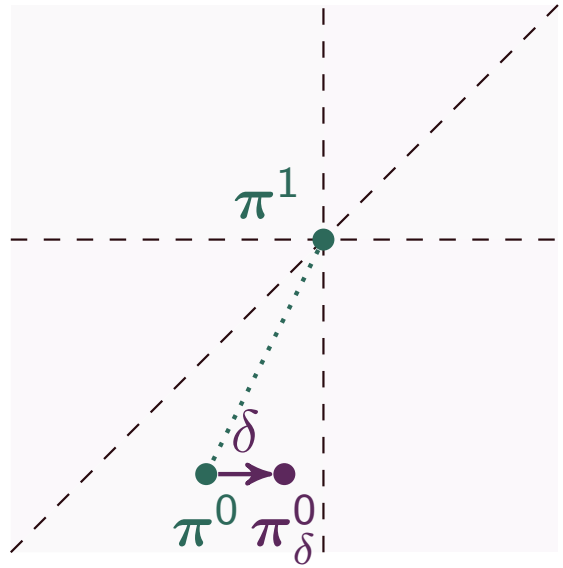
Problem: multiple boundaries intersect in π^1



Degeneracy

Problem: multiple boundaries intersect in $\boldsymbol{\pi}^1$

- perturb the previous potential by $\boldsymbol{\delta} := (\delta^n, \delta^{n-1}, \ldots, \delta)$ for some $\delta > 0$

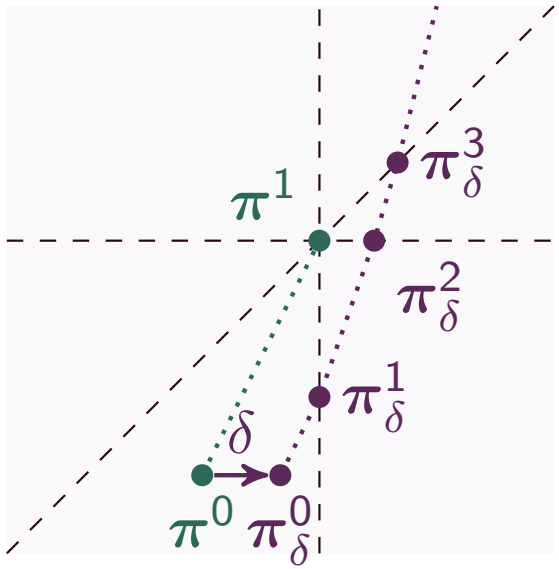


Degeneracy

Problem: multiple boundaries intersect in $\boldsymbol{\pi}^1$

- perturb the previous potential by $\boldsymbol{\delta} := (\delta^n, \delta^{n-1}, \ldots, \delta)$ for some $\delta > 0$
- track the perturbed solution curve until it moves away from all intersecting boundaries

$$\boldsymbol{\pi}^i_\delta = \boldsymbol{\pi}^1 + \mathbf{M}^i \, \boldsymbol{\delta} \text{ for } i \geq 1$$

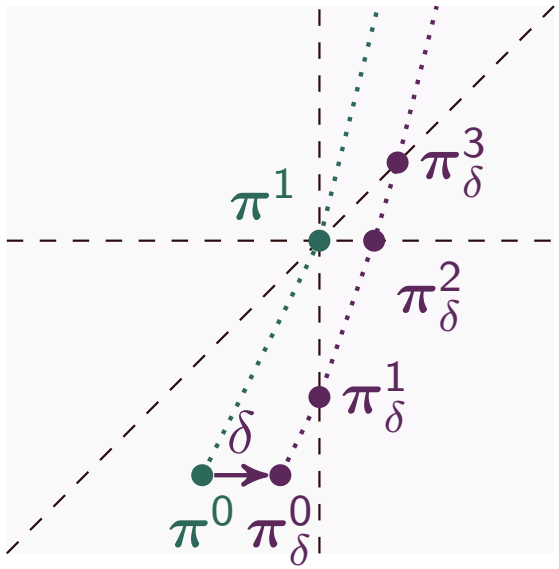


Degeneracy

Problem: multiple boundaries intersect in $\boldsymbol{\pi}^1$

- perturb the previous potential by $\boldsymbol{\delta} := (\delta^n, \delta^{n-1}, \ldots, \delta)$ for some $\delta > 0$
- track the perturbed solution curve until it moves away from all intersecting boundaries

$$\boldsymbol{\pi}^i_\delta = \boldsymbol{\pi}^1 + \mathbf{M}^i \, \boldsymbol{\delta} \text{ for } i \geq 1$$

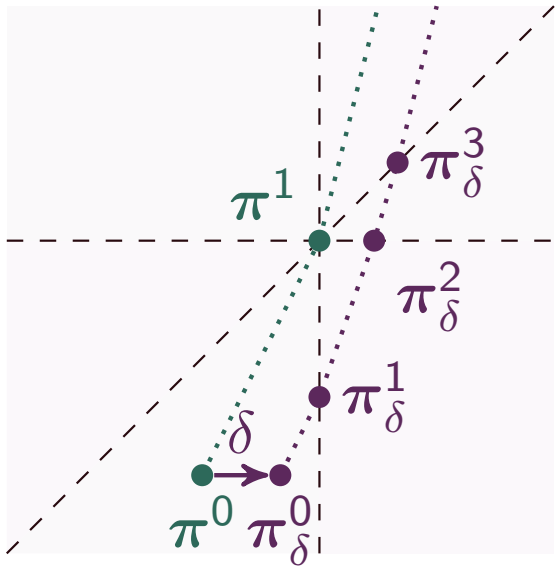


Degeneracy

Problem: multiple boundaries intersect in $\boldsymbol{\pi}^1$

- perturb the previous potential by $\boldsymbol{\delta} := (\delta^n, \delta^{n-1}, \ldots, \delta)$ for some $\delta > 0$
- track the perturbed solution curve until it moves away from all intersecting boundaries

$$\boldsymbol{\pi}^i_\delta = \boldsymbol{\pi}^1 + \mathbf{M}^i \, \boldsymbol{\delta} \text{ for } i \geq 1$$



- crossed boundary e minimizes

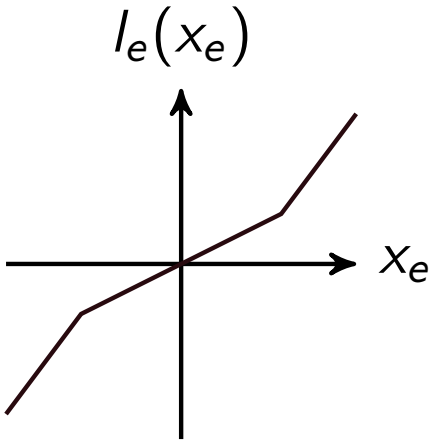
$$\epsilon^i_\delta(e) = \mathbf{m}^\top_{e,i} \, \boldsymbol{\delta} \quad \text{with} \quad \mathbf{m}^\top_{e,i} \, \boldsymbol{\delta} > 0$$

for δ small enough this is equivalent to finding the lexicographically smallest vector $\mathbf{m}^\top_{e,i}$.

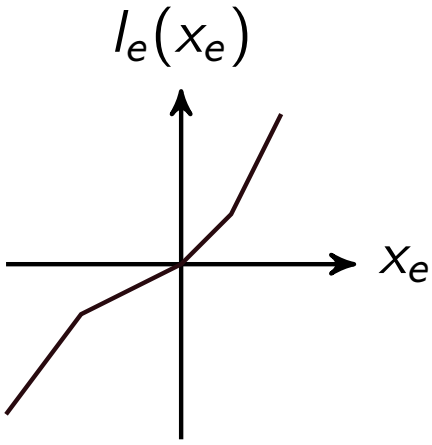
Discontinuous Cost Functions

Why Care About Discontinuities?

continuous functions



- point-symmetric

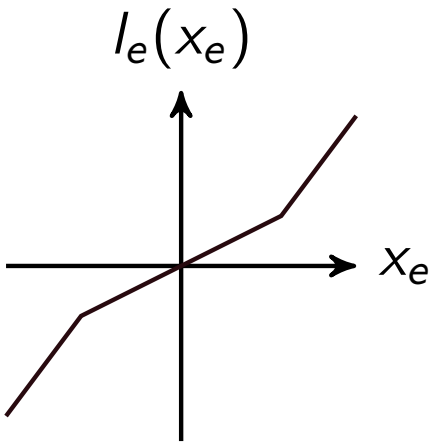


- non-symmetric

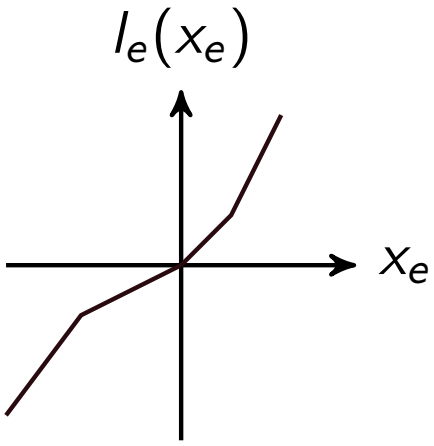
discontinuous functions

Why Care About Discontinuities?

continuous functions

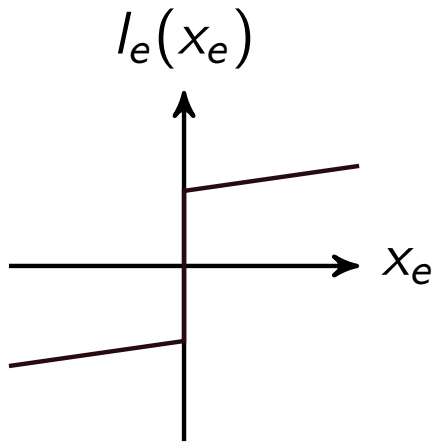


- point-symmetric



- non-symmetric

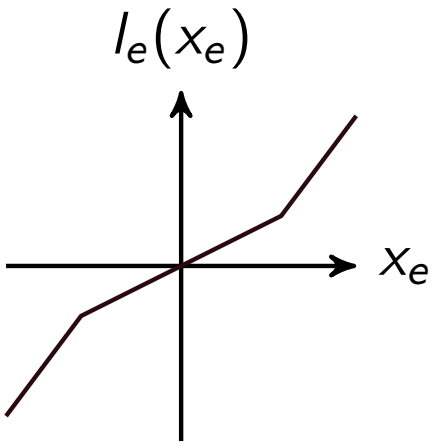
discontinuous functions



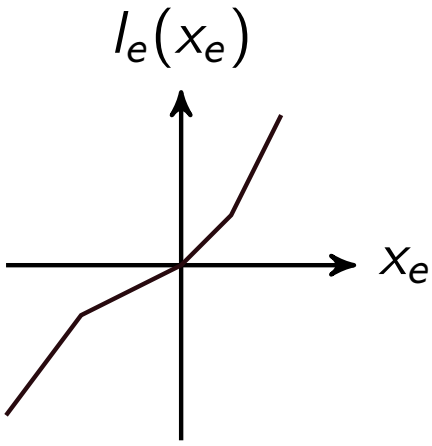
- affine

Why Care About Discontinuities?

continuous functions

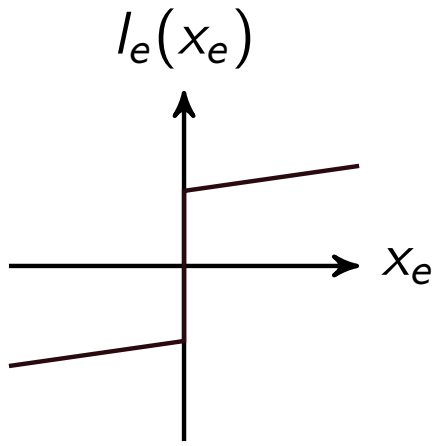


- point-symmetric

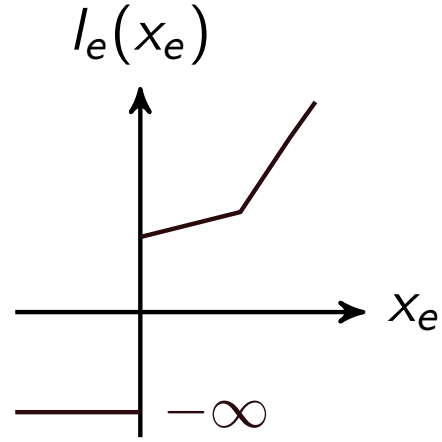


- non-symmetric

discontinuous functions



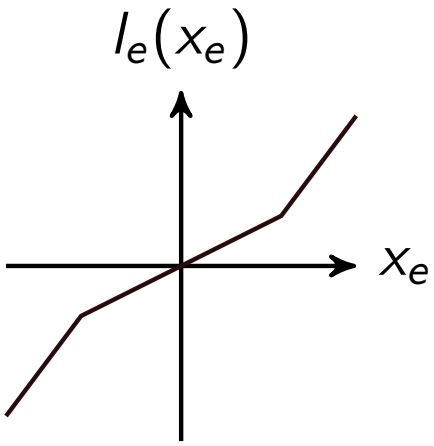
- affine



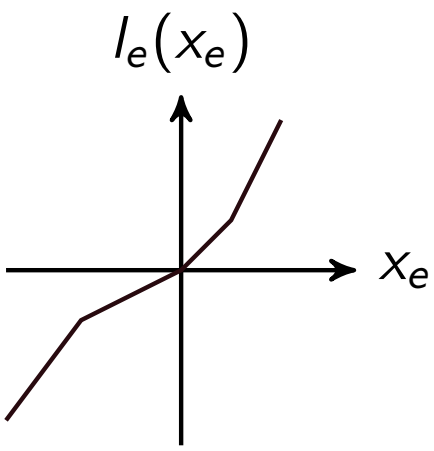
- directed edge

Why Care About Discontinuities?

continuous functions

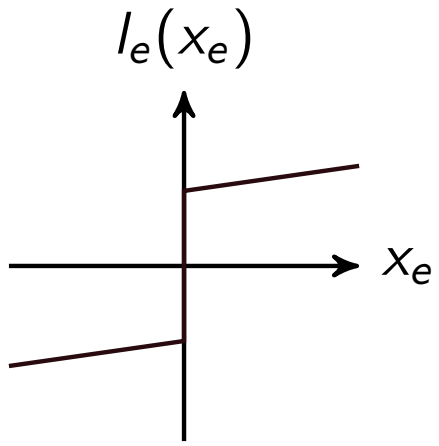


- point-symmetric

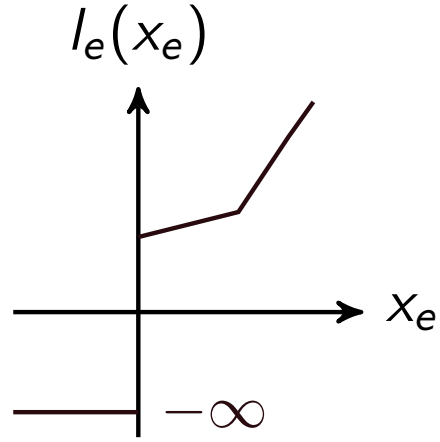


- non-symmetric

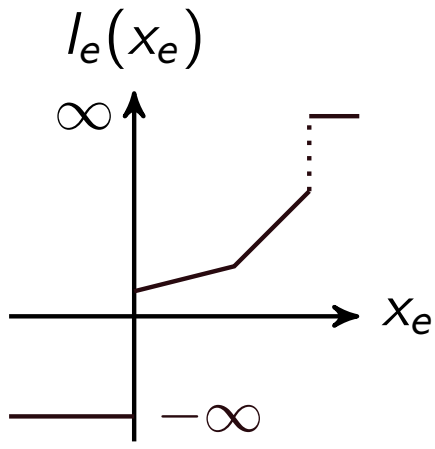
discontinuous functions



- affine



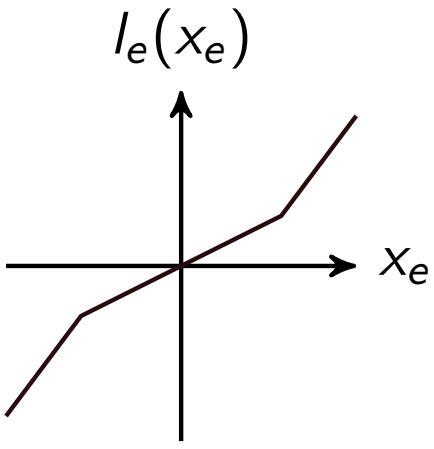
- directed edge



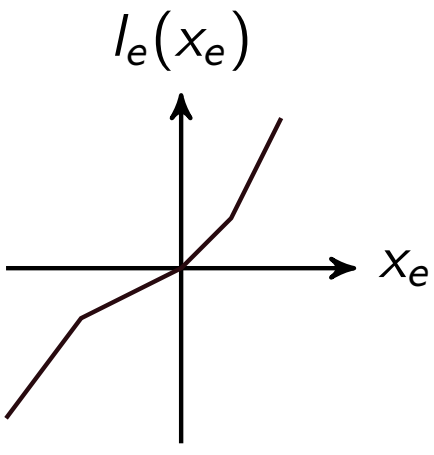
- edge capacity

Why Care About Discontinuities?

continuous functions

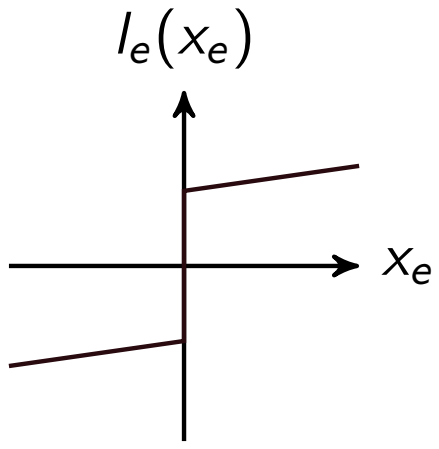


- point-symmetric

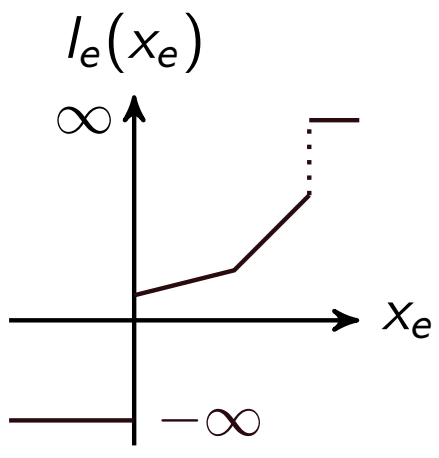


- non-symmetric

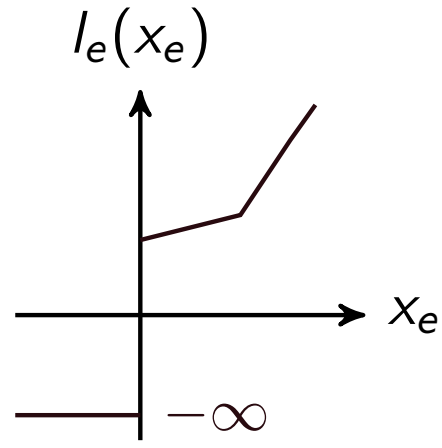
discontinuous functions



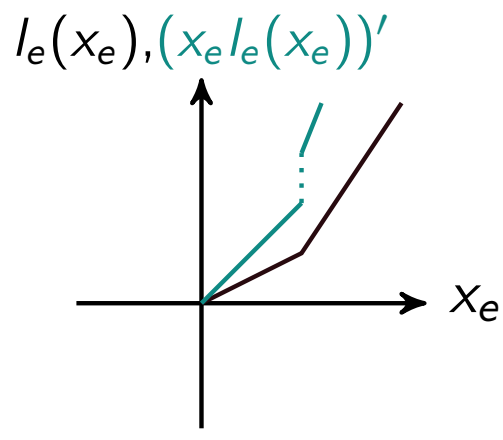
- affine



- edge capacity

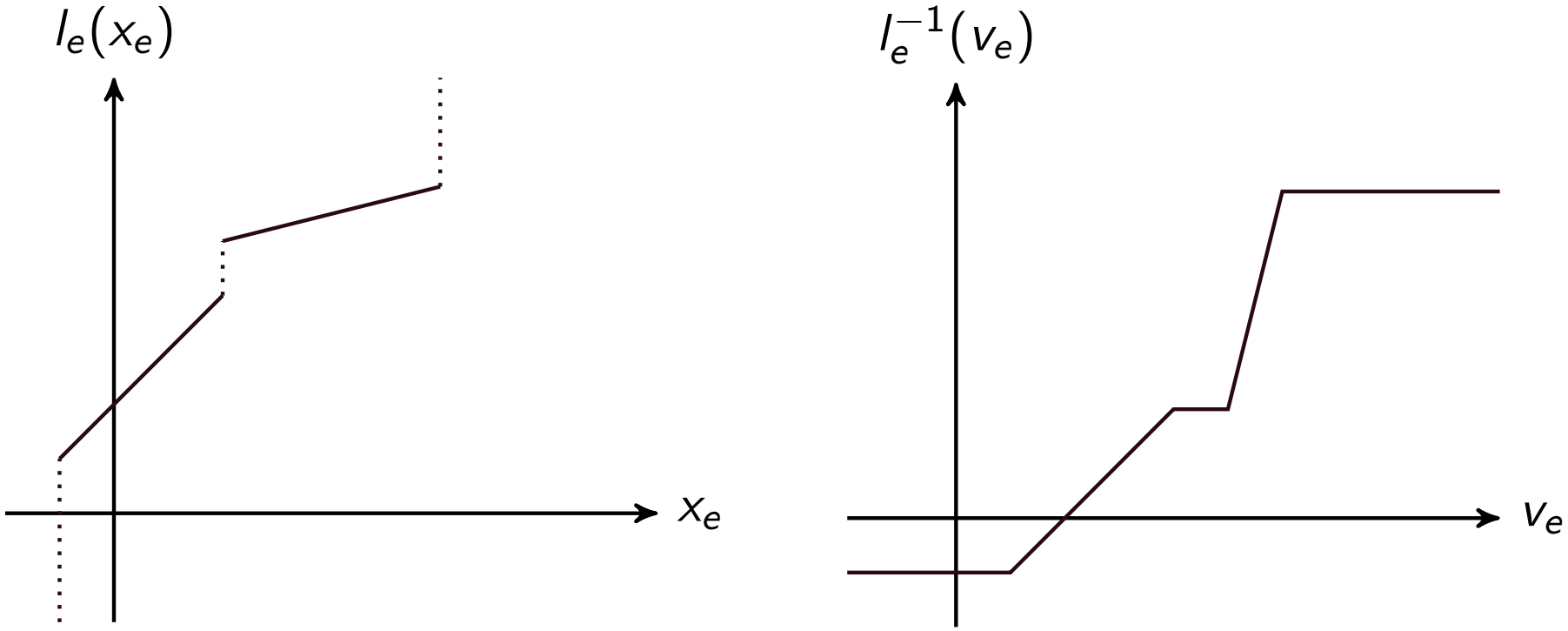


- directed edge



- marginal costs

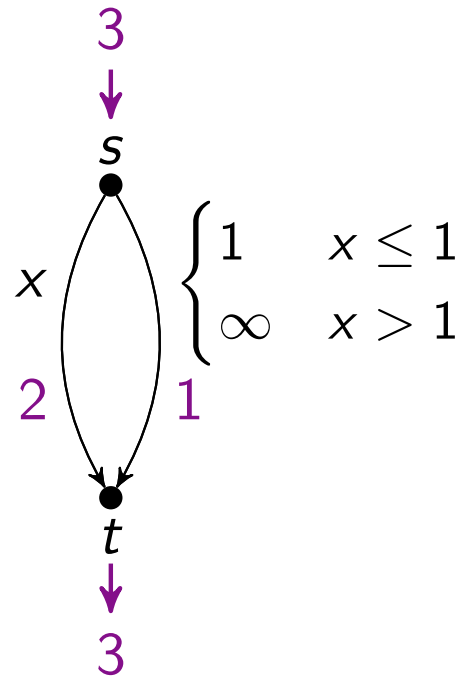
Discontinuous Cost functions



- inverse cost function is still continuous, but has constant parts

From Wardrop Equilibrium to User Equilibrium

- networks with discontinuous costs need not possess a Wardrop Equilibrium
- use concept of a User Equilibrium instead [Dafermos 1971; Bernstein and Smith 1994; Correa et al. 2004]



Definition

A feasible flow $\mathbf{x}$ is a User Equilibrium (UE) if

$$x_P > 0 \;\Rightarrow\; \liminf_{\epsilon \downarrow 0} l_P(\mathbf{x}) \leq l_Q(\mathbf{x} + \epsilon\boldsymbol{\chi}_Q - \epsilon\boldsymbol{\chi}_P) \qquad \forall P, Q \in \mathcal{P}.$$

Characterization of User Equilibria

Assume

- l_e non-negative and lower semi-continuous on $\mathbb{R}_{>0}$
- l_e non-positive and upper semi-continuous on $\mathbb{R}_{<0}$

Lemma

A feasible flow $\mathbf{x}$ is a UE iff there is a vector $\boldsymbol{\pi} \in \mathbb{R}^V$ such that

$$\liminf_{x \uparrow x_e} l_e(x) \leq \boldsymbol{\gamma}_e^\top \boldsymbol{\pi} \leq \liminf_{x \downarrow x_e} l_e(x) \qquad \forall e \in E.$$

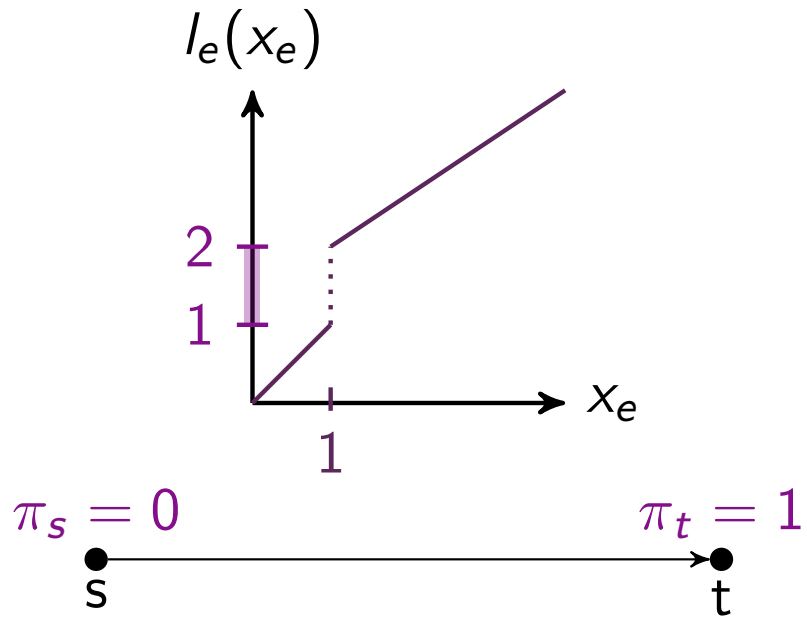
Ambiguous Regions

- excess still linear:

$$\mathbf{y} = \mathbf{\Gamma}\,\mathbf{C_t}\,\mathbf{\Gamma}^\top\,\boldsymbol{\pi} + \mathbf{\Gamma}\,\mathbf{d_t} = \mathbf{L_t}\,\boldsymbol{\pi} + \tilde{\mathbf{d}}_\mathbf{t}$$

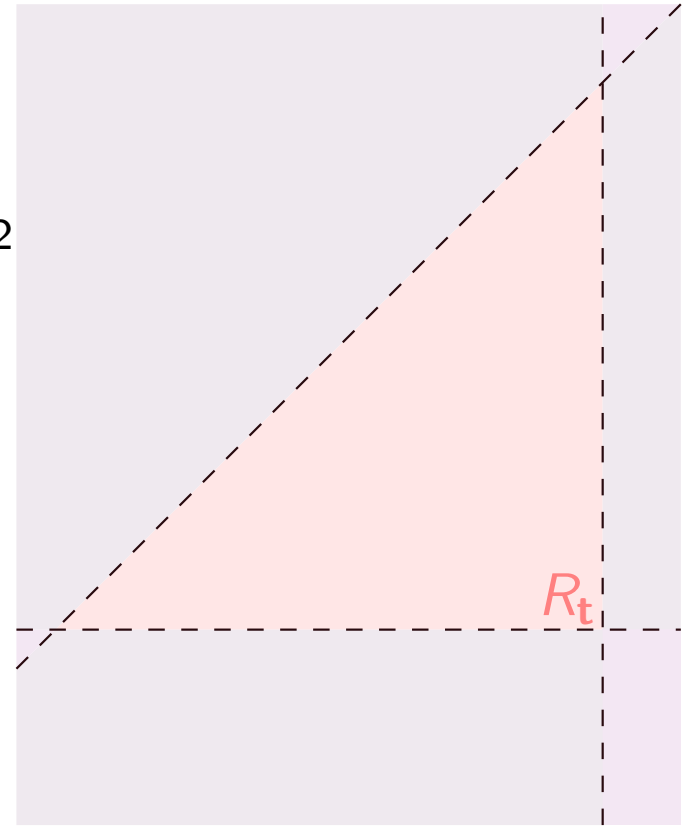
  but $\check{\mathbf{L}}_\mathbf{t}$ may be singular if too many entries of $\mathbf{C_t}$ are zero.

- **Fact:** $\check{\mathbf{L}}_\mathbf{t}$ regular $\Leftrightarrow$ all vertices connected by edges with $c_e > 0$.



Ambiguous Regions

- **Fact:** $\check{\mathbf{L}}_\mathbf{t}$ singular
  $\Rightarrow$ graph decomposes in connected components $\mathcal{C}_1, \mathcal{C}_2$ with $s \in \mathcal{C}_1$ and $t \in \mathcal{C}_2$



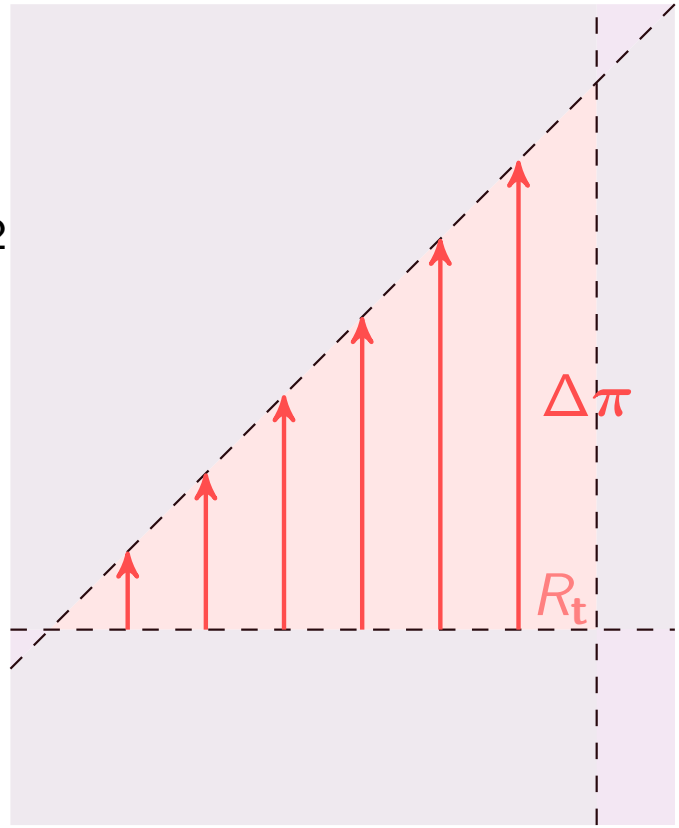
Ambiguous Regions

- **Fact:** $\check{\mathbf{L}}_\mathbf{t}$ singular
  $\Rightarrow$ graph decomposes in connected components $\mathcal{C}_1, \mathcal{C}_2$ with $s \in \mathcal{C}_1$ and $t \in \mathcal{C}_2$
- direction vector $\Delta\boldsymbol{\pi}$ with

$$\Delta\pi_v = \begin{cases} 1 & v \in \mathcal{C}_2 \\ 0 & v \in \mathcal{C}_1 \end{cases}$$

- flow is constant along $\Delta\boldsymbol{\pi}$, i.e.

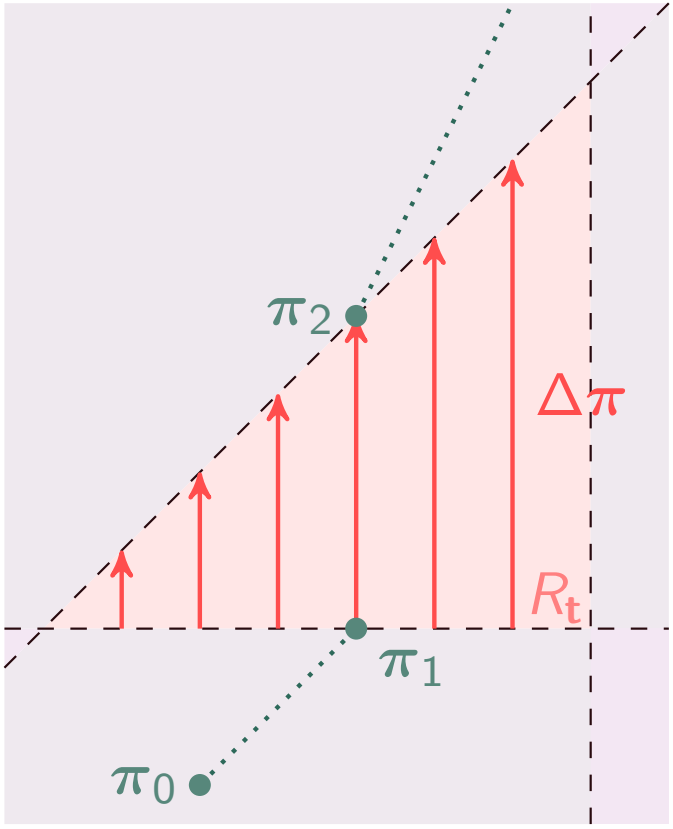
$$\begin{aligned} \mathbf{x} &= \mathbf{C_t}\,\boldsymbol{\Gamma}^\top \boldsymbol{\pi} + \mathbf{d_t} \\ &= \mathbf{C_t}\,\boldsymbol{\Gamma}^\top (\boldsymbol{\pi} + \Delta\boldsymbol{\pi}) + \mathbf{d_t} \end{aligned}$$



Ambiguous Regions

- **Fact:** $\check{\mathbf{L}}_\mathbf{t}$ singular
  $\Rightarrow$ graph decomposes in connected components $\mathcal{C}_1, \mathcal{C}_2$ with $s \in \mathcal{C}_1$ and $t \in \mathcal{C}_2$
- direction vector $\Delta\boldsymbol{\pi}$ with

$$\Delta\pi_v = \begin{cases} 1 & v \in \mathcal{C}_2 \\ 0 & v \in \mathcal{C}_1 \end{cases}$$



- flow is constant along $\Delta\boldsymbol{\pi}$, i.e.

$$\begin{aligned} \mathbf{x} &= \mathbf{C_t}\,\boldsymbol{\Gamma}^\top \boldsymbol{\pi} + \mathbf{d_t} \\ &= \mathbf{C_t}\,\boldsymbol{\Gamma}^\top (\boldsymbol{\pi} + \Delta\boldsymbol{\pi}) + \mathbf{d_t} \end{aligned}$$

- solution curve moves along $\Delta\boldsymbol{\pi}$ to the next boundary

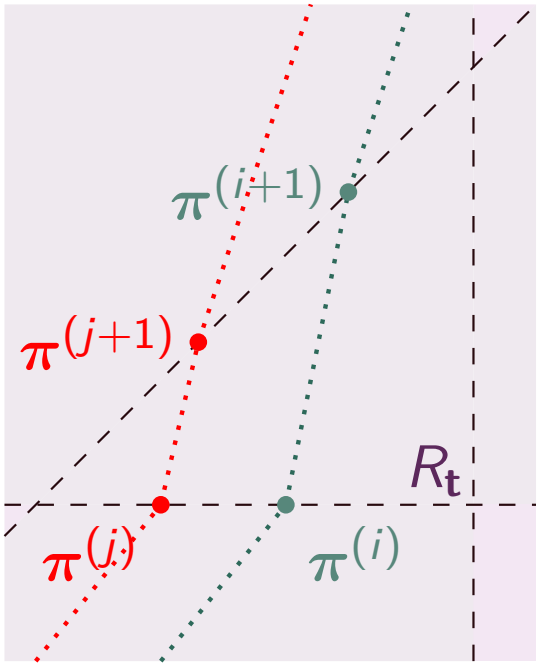
Complexity

Theorem

The algorithm visits each region $R_{\mathbf{t}}$ at most once.

Proof.

- let $\mathbf{t} = \mathbf{t}^{(i)} = \mathbf{t}^{(j)}$ be two identical region vectors with $i < j$
- $i + 1 < j$ since boundary crossing well-defined
- $\boldsymbol{\pi}^{(i+1)} = \boldsymbol{\pi}^{(i)} + \epsilon^{(i)} \Delta\boldsymbol{\pi}^{(i)}$ on boundary of $R_{\mathbf{t}}$
- $\boldsymbol{\pi}^{(i)} + \lambda\, \Delta\boldsymbol{\pi}^{(i)} \notin R_{\mathbf{t}}$ for all $\lambda > \epsilon_i = \lambda_{i+1} - \lambda_i$ since $R_{\mathbf{t}}$ is convex



Proof (continued)

Recall: $\boldsymbol{\pi}^{(i)} + \lambda\,\Delta\boldsymbol{\pi}^{(i)} \notin R_{\mathbf{t}}$ for all $\lambda > \epsilon_i = \lambda_{i+1} - \lambda_i$ $\quad(\star)$

- by definition
$$\lambda_i\,\Delta\mathbf{y} = \mathbf{L_t}\boldsymbol{\pi}^{(i)} - \tilde{\mathbf{d}}_{\mathbf{t}}$$
$$\lambda_j\,\Delta\mathbf{y} = \mathbf{L_t}\boldsymbol{\pi}^{(j)} - \tilde{\mathbf{d}}_{\mathbf{t}}$$
- we obtain
$$(\lambda_j - \lambda_i)\Delta\hat{\mathbf{y}} = \hat{\mathbf{L}}_{\mathbf{t}}(\hat{\boldsymbol{\pi}}^{(j)} - \hat{\boldsymbol{\pi}}^{(i)})$$
$$\Leftrightarrow \quad (\lambda_j - \lambda_i)\hat{\mathbf{L}}_{\mathbf{t}}^{-1}\Delta\hat{\mathbf{y}} = (\hat{\boldsymbol{\pi}}^{(j)} - \hat{\boldsymbol{\pi}}^{(i)})$$
$$\Leftrightarrow \quad (\lambda_j - \lambda_i)\Delta\hat{\boldsymbol{\pi}}^{(\mathbf{t})} = (\hat{\boldsymbol{\pi}}^{(j)} - \hat{\boldsymbol{\pi}}^{(i)})$$
- thus $\boldsymbol{\pi}^{(j)} = \boldsymbol{\pi}^{(i)} + (\lambda_j - \lambda_i)\Delta\boldsymbol{\pi}^{(\mathbf{t})}$
- since $\boldsymbol{\pi}^{(j)}, \boldsymbol{\pi}^{(i)} \in R_{\mathbf{t}}$ and $R_{\mathbf{t}}$ convex, also $\boldsymbol{\pi}^{(i)} + \lambda\Delta\boldsymbol{\pi}^{(\mathbf{t})} \in R_{\mathbf{t}}$ for all $\lambda \in [0, \lambda_j - \lambda_i]$
- contradiction to $(\star)$ □

Runtime

- algorithm finite in case of finitely many breakpoints
- without degenerate points the algorithm is output polynomial
  - polynomial in the number of segments of $x_e(r)$
  - number of segments of $x_e(r)$ can be exponential in the size of G (nested Braess' Pardox)
  - in series-parallel networks the algorithm runs in polynomial time in the input size
    (proof via inverse positivity of $\mathbf{L_t}$)
- for special case of constant cost functions with capacities, algorithm equals successive shortest path algorithm
  - exponential runtime [Zadeh 1973]
  - NP-mightyness [Disser, Skutella 2015]
- output polynomial runtime with degenerate points can be achieved via convex programming

Directions via Convex Program

- let $\mathbf{x}$ be a Wardrop equilibrium for some rate $q > 0$
- let $a_e = \partial^+ l_e(x_e)/\partial x_e$, $a_e = \partial^+ l_e(x_e)/\partial x_e$, $\mathbf{A}^+ = \operatorname{diag}(a^+_{e_1}, \ldots, a^+_{e_m})$, $\mathbf{A}^- = \operatorname{diag}(a^-_{e_1}, \ldots, a^-_{e_m})$
- consider the quadratic program

$$\begin{aligned} \min \quad & \frac{1}{2}(\mathbf{z}^+)^\top \mathbf{A}^+ \mathbf{z}^+ + \frac{1}{2}(\mathbf{z}^-)^\top \mathbf{A}^- \mathbf{z}^- \\ \text{s.t.:} \quad & \mathbf{\Gamma z} = \chi_s - \chi_t \\ & \mathbf{z} = \mathbf{z}^+ + \mathbf{z}^- \\ & \mathbf{z}^+ \geq 0 \\ & \mathbf{z}^- \geq 0 \end{aligned} \tag{QP}$$

Directions via Convex Program (continued)

Fact: (QP) can be solved exactly in polynomial time.

[Kozlov, Tarasov, Khachiyan 1980]

Theorem

There is $\epsilon > 0$ such that $\mathbf{x} + \lambda \mathbf{z}$ is a WE with rate $q + \lambda$ for all $\lambda \in [0, \epsilon]$.

Proof idea.

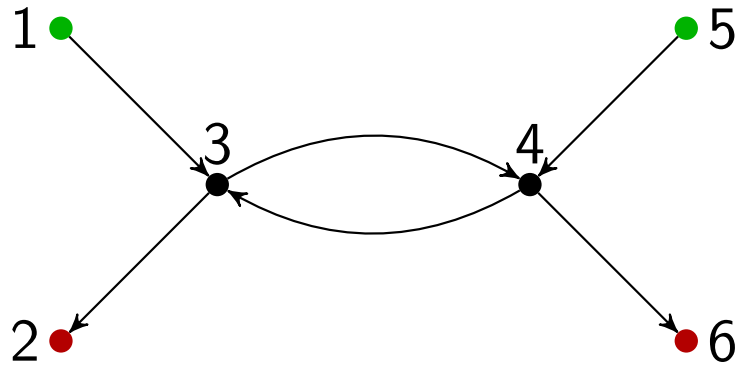
- use KKT conditions of the program together with the potential formulation of a WE

Multiple Sources and Sinks

Multiple Sources and Sinks

- all results hold for arbitrary excess directions $\Delta\mathbf{y}$
- multiple
  - sources (vertices with $\Delta y_v < 0$)
  - sinks (vertices with $\Delta y_v > 0$)

  are possible

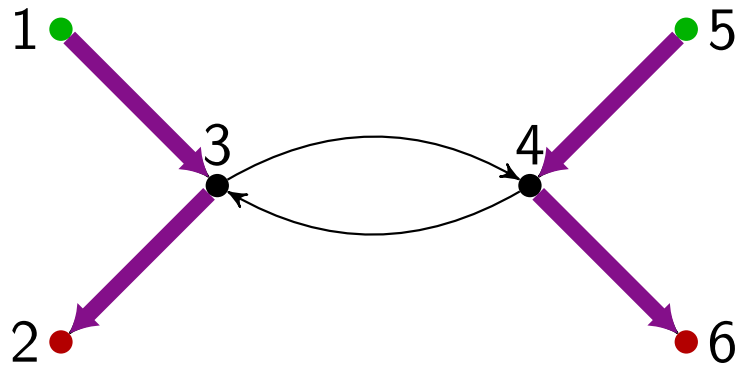


$$\Delta\mathbf{y} = \begin{pmatrix} -1 \\ 1 \\ 0 \\ 0 \\ -1 \\ 1 \end{pmatrix}$$

Multiple Sources and Sinks

- all results hold for arbitrary excess directions $\Delta\mathbf{y}$
- multiple
  - sources (vertices with $\Delta y_v < 0$)
  - sinks (vertices with $\Delta y_v > 0$)

  are possible
- Flows may cancel out $\rightarrow$ only one commodity



$$\Delta\mathbf{y} = \begin{pmatrix} -1 \\ 1 \\ 0 \\ 0 \\ -1 \\ 1 \end{pmatrix}$$

Multiple Commodities

Lemma

A multi-commodity flow $\mathbf{X} = (x_{e,k})_{e \in E, k \in [K]}$ is a WE if and only if there are vectors $\pi^k \in \mathbb{R}^V$ for all $k \in [K]$ such that

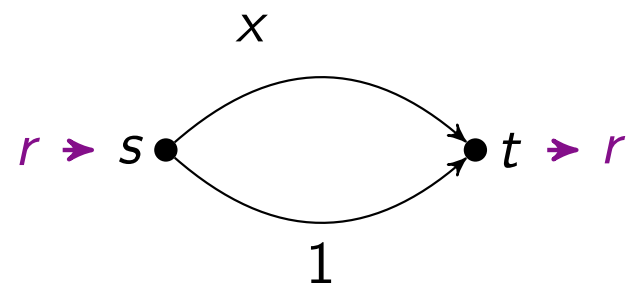
$$\gamma_e^\top \pi^k \leq l_e(z_e) = a_e z_e + b_e, \qquad \forall e \in E$$
$$\gamma_e^\top \pi^k = l_e(z_e) = a_e z_e + b_e \qquad \forall e \in E : x_{k,e} > 0,$$

where $z_e := \sum_{k=1}^{K} x_{e,k}$.

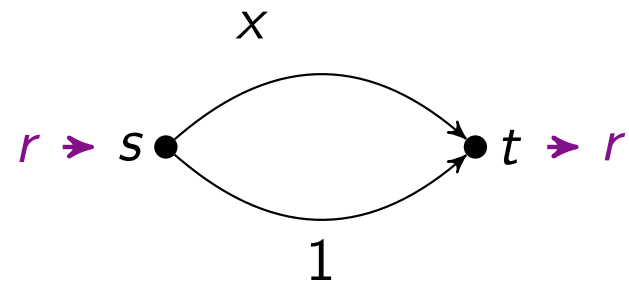
- every commodity k has its own potential π^k
- pivoting approach infeasible
- inside a region a direction vector $\Delta\pi$ can be computed by solving a convex program similar to (QP)

Edge Pricing with Unknown Demands

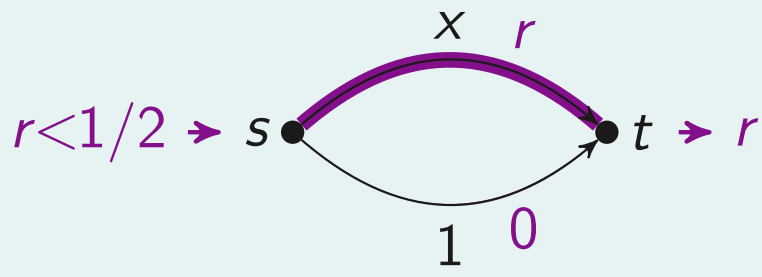
Pigou's Network

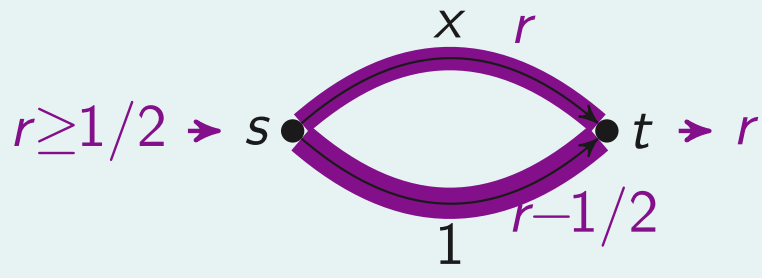


Pigou's Network



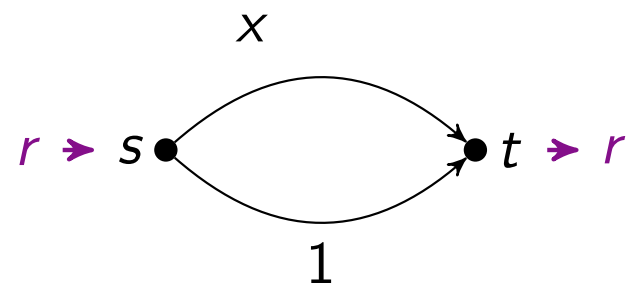
System optimum



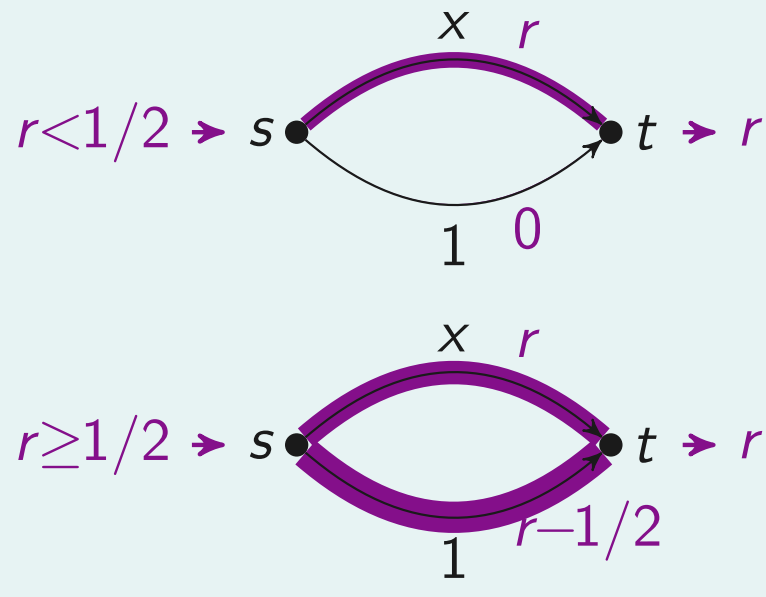


Pigou's Network

- marginal cost tolls
  $\tau_e^* = l_e'(x_e^*(r))$ yield optimum
  [Beckman et al. 1956]

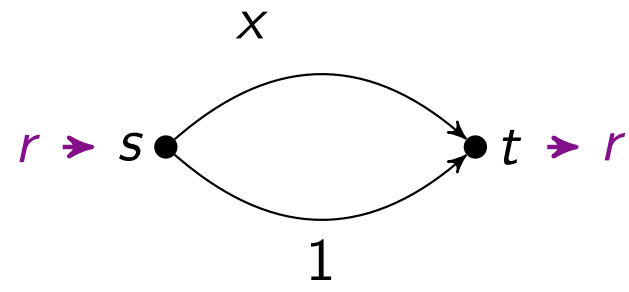


System optimum

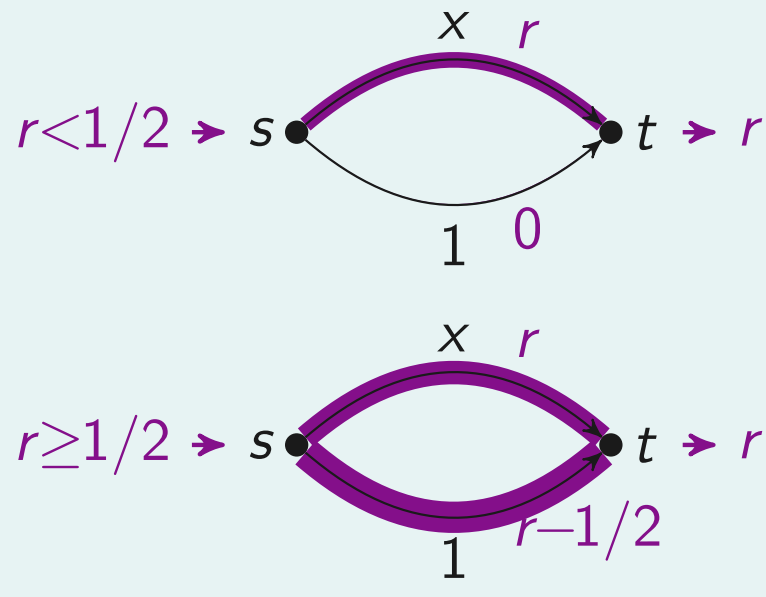


Pigou's Network

- marginal cost tolls
  $\tau_e^* = l_e'(x_e^*(r))$ yield optimum
  [Beckman et al. 1956]
- marginal cost tolls depend on r

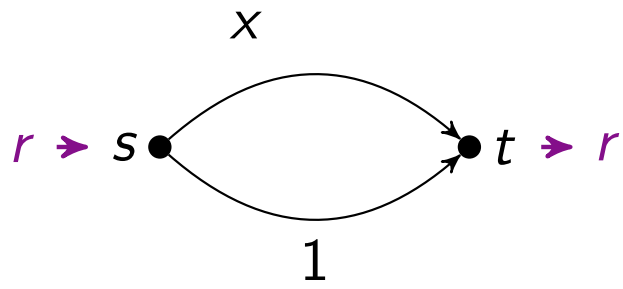


System optimum

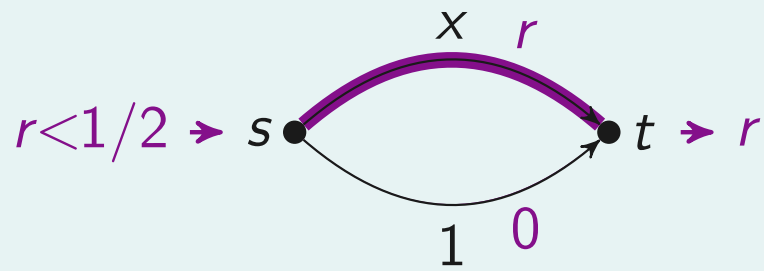


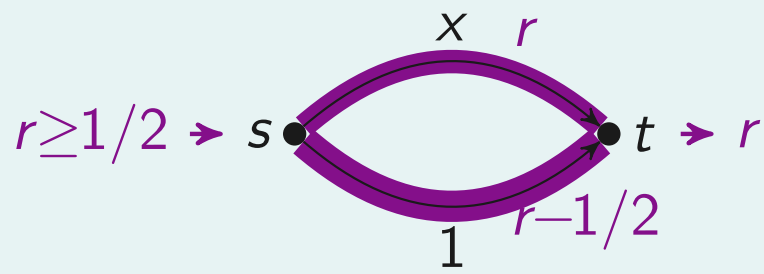
Pigou's Network

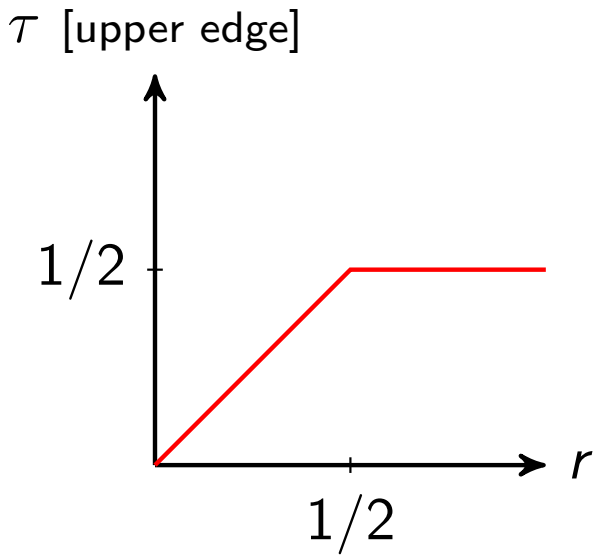
- marginal cost tolls $\tau_e^* = l_e'(x_e^*(r))$ yield optimum [Beckman et al. 1956]
- marginal cost tolls depend on r
- misperception of r leads to suboptimal toll



System optimum

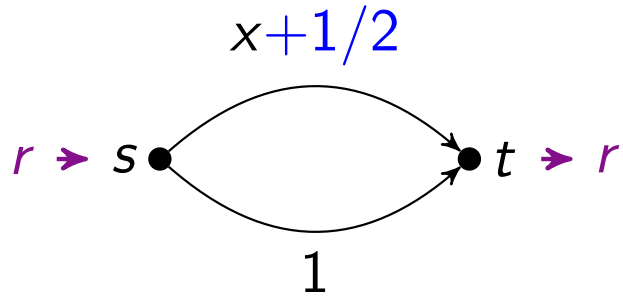




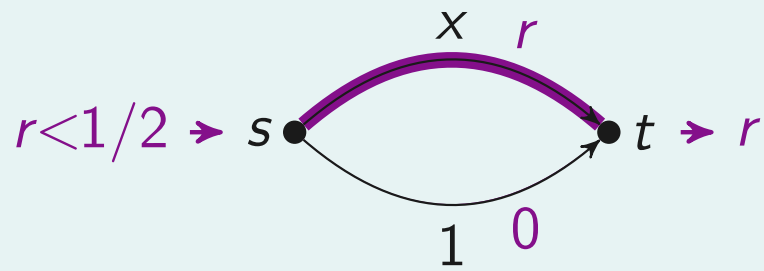


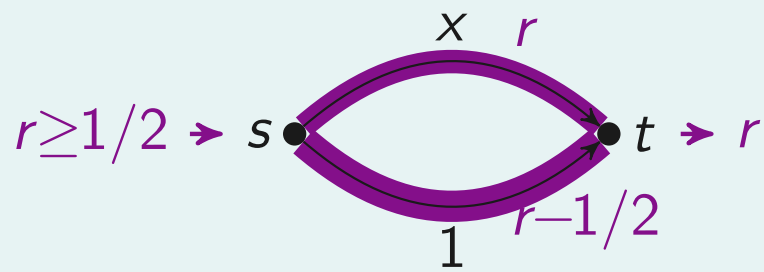
Pigou's Network

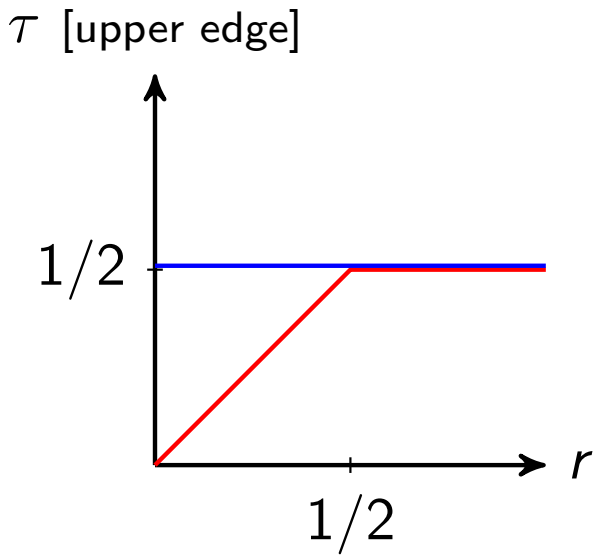
- marginal cost tolls $\tau_e^* = l_e'(x_e^*(r))$ yield optimum [Beckman et al. 1956]
- marginal cost tolls depend on r
- misperception of r leads to suboptimal toll
- demand-independent optimal toll $\bar{\tau} = 1/2$



System optimum

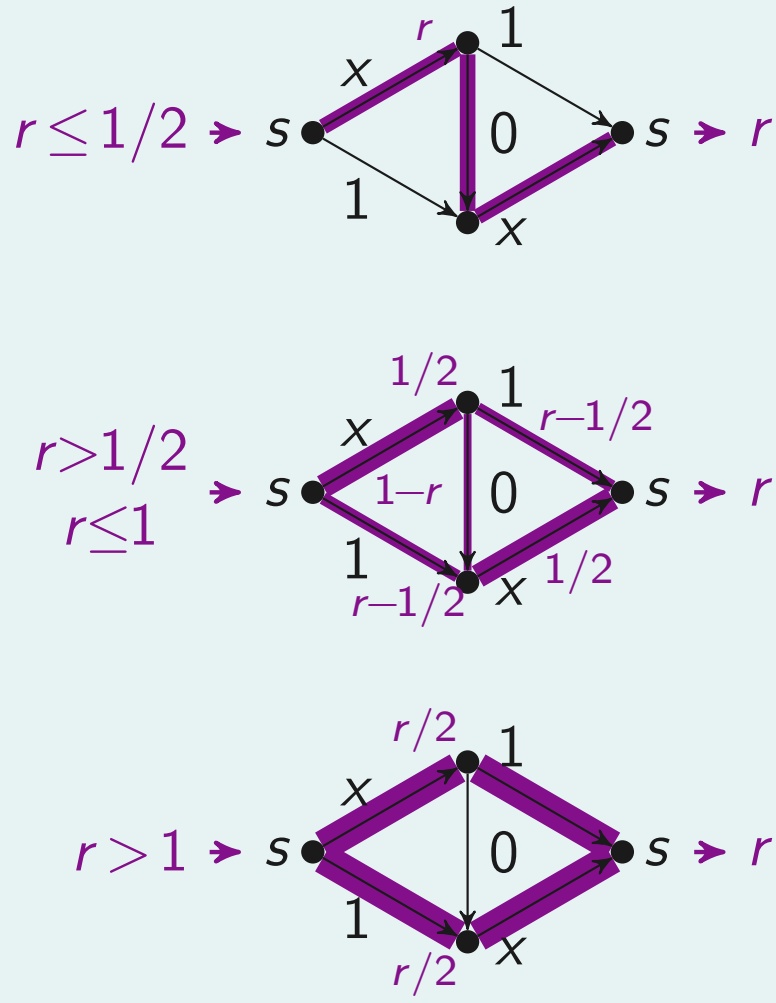






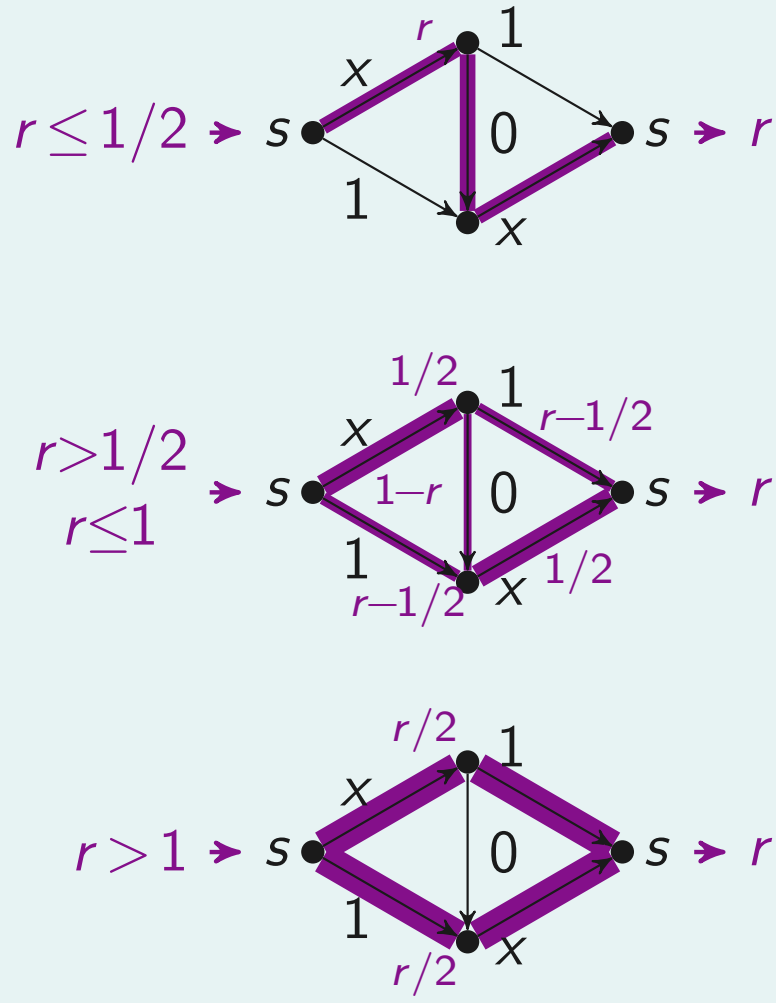
Braess' Network

System optimum



Braess' Network

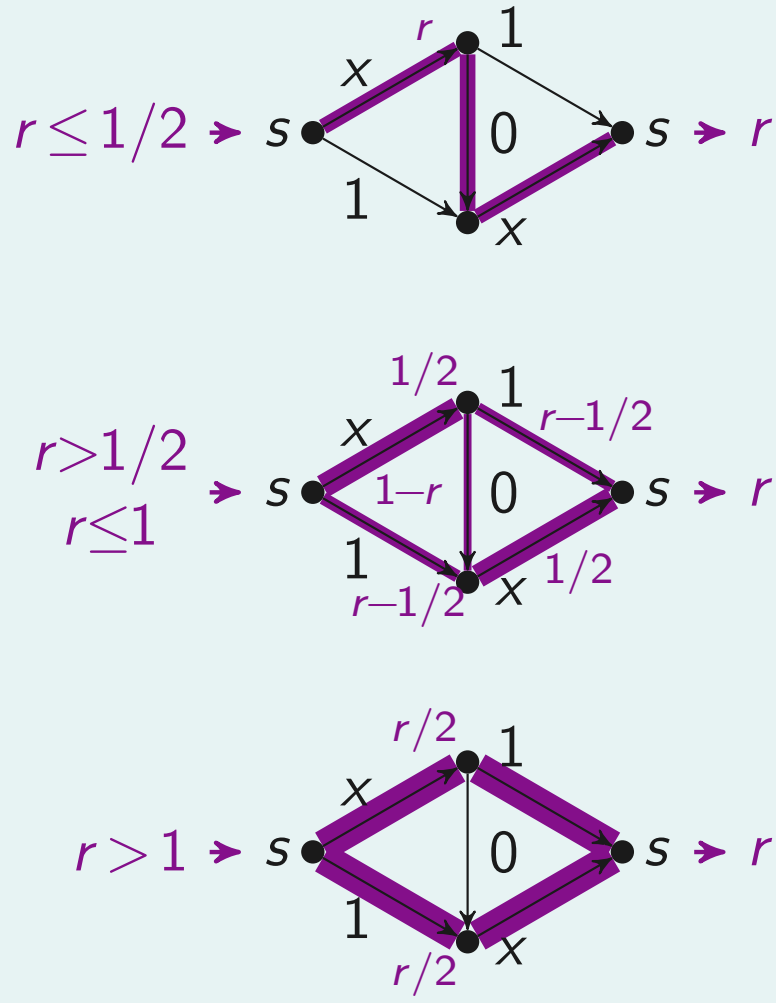
System optimum



Braess' Network

- marginal cost tolls $\tau_e^* = l_e'(x_e^*(r))$ yield optimum [Beckman et al. 1956]
- marginal cost tolls depend on r
- marginal cost pricing harmful for small flow values

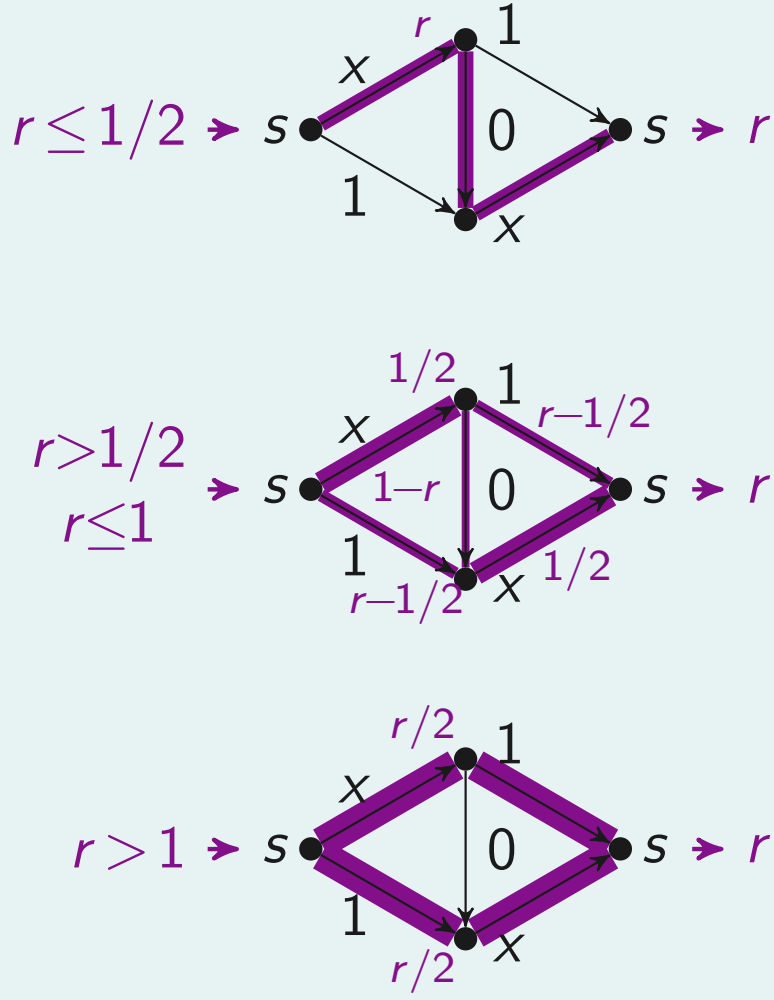
System optimum



Braess' Network

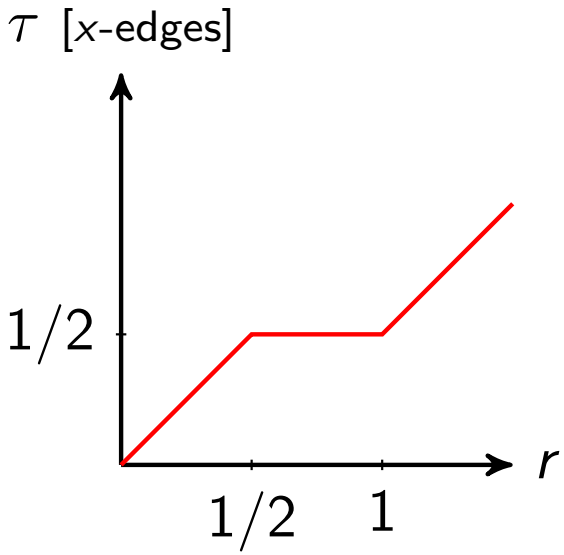
- marginal cost tolls $\tau_e^* = l_e'(x_e^*(r))$ yield optimum [Beckman et al. 1956]
- marginal cost tolls depend on r
- marginal cost pricing harmful for small flow values

System optimum

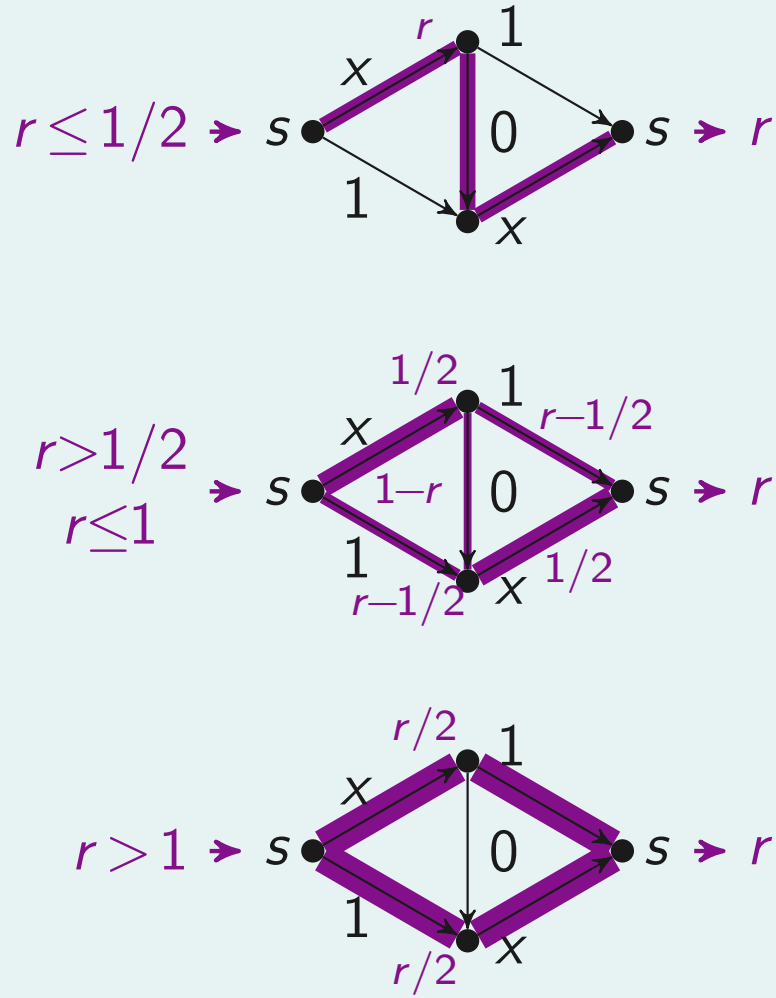


Braess' Network

- marginal cost tolls $\tau_e^* = l_e'(x_e^*(r))$ yield optimum [Beckman et al. 1956]
- marginal cost tolls depend on r
- marginal cost pricing harmful for small flow values

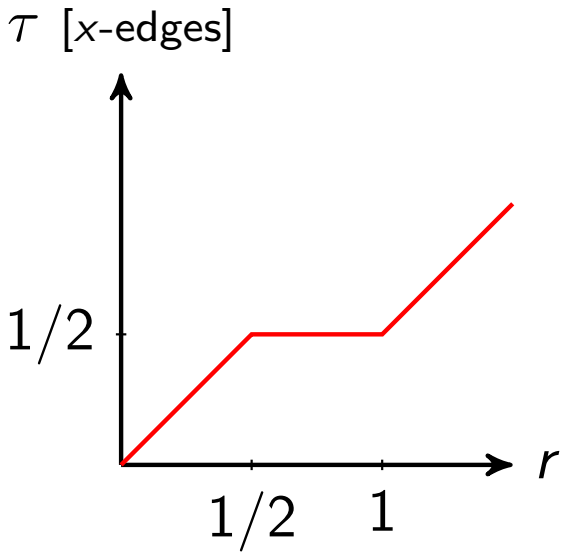


System optimum

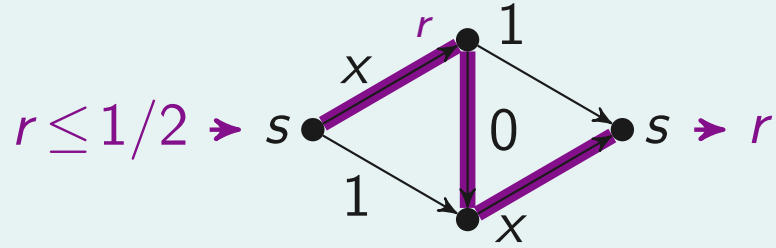


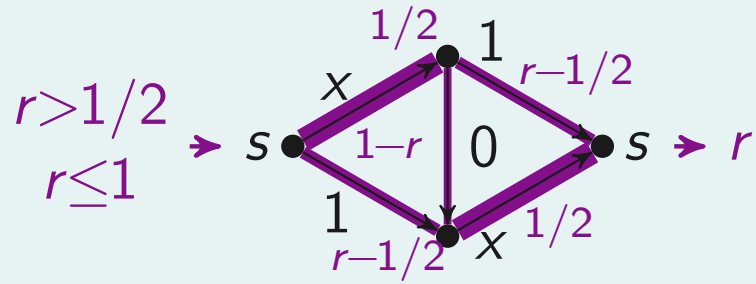
Braess' Network

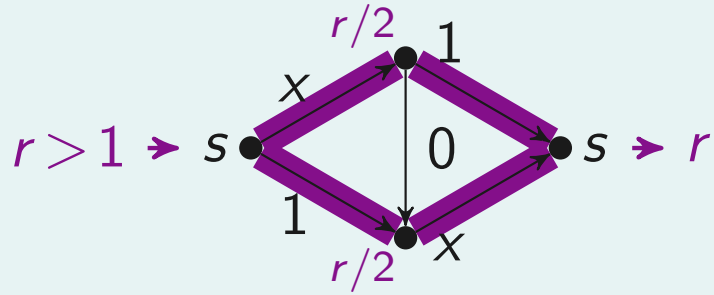
- marginal cost tolls $\tau_e^* = l_e'(x_e^*(r))$ yield optimum [Beckman et al. 1956]
- marginal cost tolls depend on r
- marginal cost pricing harmful for small flow values
- demand-independent optimal toll $\bar{\tau} = 1/2$ on 0-edge



System optimum







Characterization of demand-independent tolls

Theorem [Colini-Baldeschi, K., Scarsini ICALP 2018]

Demand-independent tolls exist if and only if all cost functions are of BPR type $l_e = a_e x^t + b_e$.

- includes functions used by US Bureau of Public Roads for $t = 4$

Characterization of demand-independent tolls

Theorem [Colini-Baldeschi, K., Scarsini ICALP 2018]

Demand-independent tolls exist if and only if all cost functions are of BPR type $l_e = a_e x^t + b_e$.

- includes functions used by US Bureau of Public Roads for $t = 4$

Theorem [Colini-Baldeschi, K., Scarsini ICALP 2018]

Demand-independent tolls are unique (except for constant shifts and paths that are never used).

Characterization of demand-independent tolls

Theorem [Colini-Baldeschi, K., Scarsini ICALP 2018]

Demand-independent tolls exist if and only if all cost functions are of BPR type $l_e = a_e x^t + b_e$.

- includes functions used by US Bureau of Public Roads for $t = 4$

Theorem [Colini-Baldeschi, K., Scarsini ICALP 2018]

Demand-independent tolls are unique (except for constant shifts and paths that are never used).

- demand-independent tolls may be negative
- non-negative demand-independent tolls exist, e.g., for acyclic graphs (e.g., Pigou's, Braess')
- aggregatively non-negative tolls exist under a weaker condition (e.g. all single-commodity networks)

Conclusions

- efficient algorithm for computing all Wardrop equilibria parametrized by the flow value
  - degenerate points
  - discontinuous cost functions
  - edge capacities
  - multi-commodity networks
- demand-independent tolls that enforce the system optimal flow
  - exist only for BPR-functions
  - essentially unique
  - additional topological properties give non-negative tolls

Thank you.