Traffic routing in congested networks

Equilibrium, Efficiency, and Dynamics

P. Mertikopoulos

French National Center for Scientific Research (CNRS)

Laboratoire d'Informatique de Grenoble

ROADEF-MODE colloquium, June 8, 2018

Outline

Background

Model and preliminaries

The PoA in practice

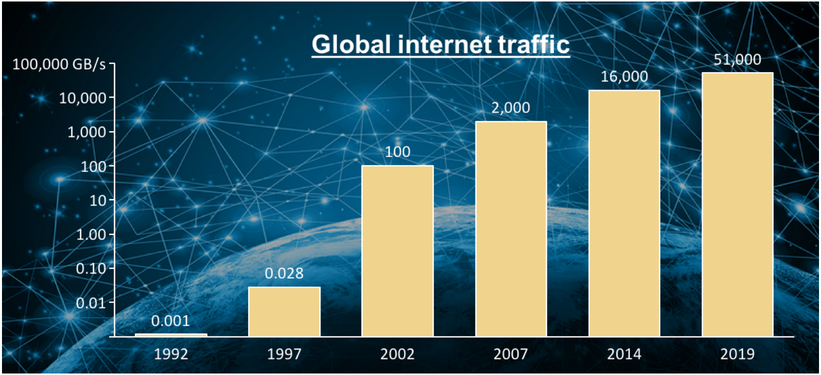
Algorithms and dynamics

Traffic...

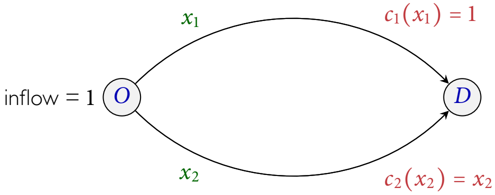
...how bad can it get?

Traffic...

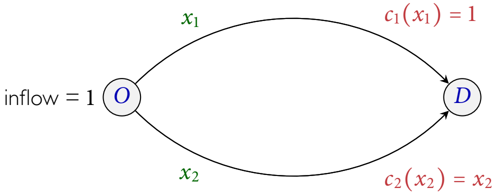
...how bad can it get?



Pigou's example (1920's)



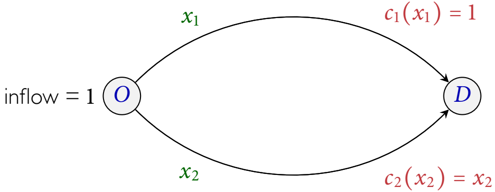
Pigou's example (1920's)



- **Equilibrium/Fair assignment**

$$c_1(x_1) = c_2(x_2) \tag{Eq}$$

Pigou's example (1920's)



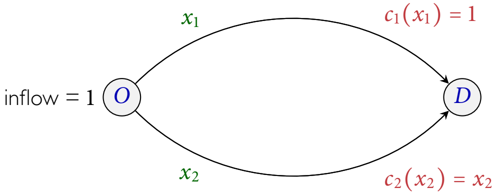
- **Equilibrium/Fair assignment**

$$c_1(x_1) = c_2(x_2) \tag{Eq}$$

- **Optimum assignment**

$$\begin{aligned} \text{minimize} \quad & C(x) = x_1 c_1(x_1) + x_2 c_2(x_2) \\ \text{subject to} \quad & x_1 + x_2 = 1 \end{aligned} \tag{Opt}$$

Pigou's example (1920's)



- Equilibrium/Fair assignment

$$c_1(x_1) = c_2(x_2) \quad \text{(Eq)}$$

- Optimum assignment

$$\begin{aligned} \text{minimize} \quad & C(x) = x_1 c_1(x_1) + x_2 c_2(x_2) \\ \text{subject to} \quad & x_1 + x_2 = 1 \end{aligned} \quad \text{(Opt)}$$

- How bad is selfish routing?

$$\text{Price of Anarchy} \equiv \frac{C(\text{Eq})}{C(\text{Opt})} = \frac{4}{3}$$

Outline

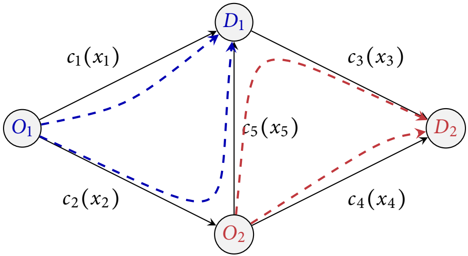
Background

Model and preliminaries

The PoA in practice

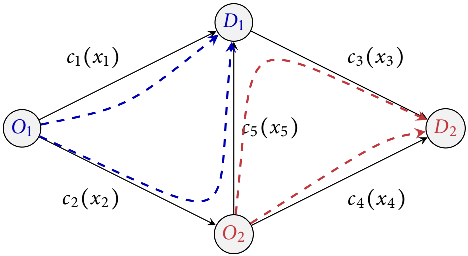
Algorithms and dynamics

The model



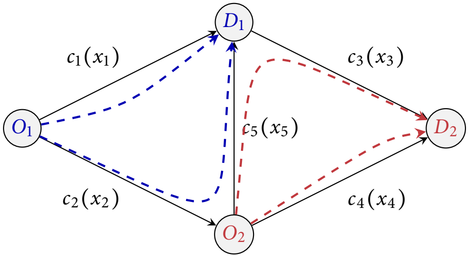
- Network: multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

The model



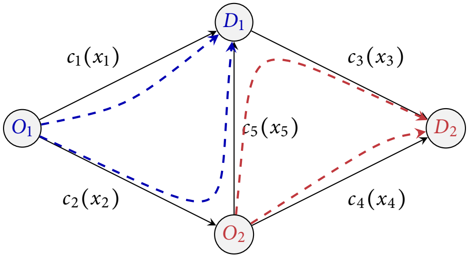
- Network: multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- O/D pairs $i \in \mathcal{I}$: origin O^i sends m^i units of traffic to destination D^i

The model



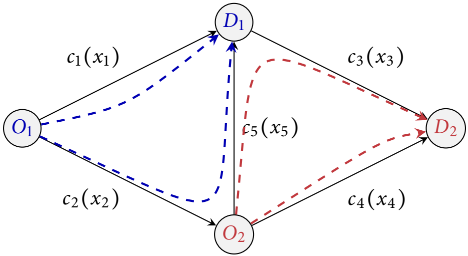
- Network: multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- O/D pairs $i \in \mathcal{I}$: origin O^i sends m^i units of traffic to destination D^i
- Paths $\mathcal{P}^i$: (sub)set of paths joining $O^i \rightsquigarrow D^i$

The model



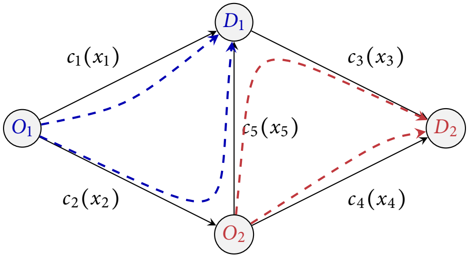
- Network: multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- O/D pairs $i \in \mathcal{I}$: origin O^i sends m^i units of traffic to destination D^i
- Paths $\mathcal{P}^i$: (sub)set of paths joining $O^i \rightsquigarrow D^i$
- Routing flow f_p: traffic along $p \in \mathcal{P} \equiv \bigcup^i \mathcal{P}^i$ generated by O/D pair owning p

The model



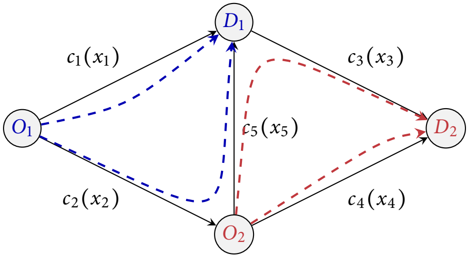
- Network: multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- O/D pairs $i \in \mathcal{I}$: origin O^i sends m^i units of traffic to destination D^i
- Paths $\mathcal{P}^i$: (sub)set of paths joining $O^i \rightsquigarrow D^i$
- Routing flow f_p: traffic along $p \in \mathcal{P} \equiv \bigcup^i \mathcal{P}^i$ generated by O/D pair owning p
- Load $x_e = \sum_{p \ni e} f_p$: total traffic along edge e

The model



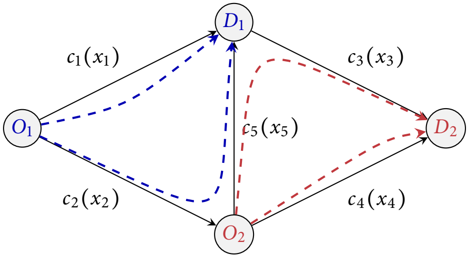
- Network: multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- O/D pairs $i \in \mathcal{I}$: origin O^i sends m^i units of traffic to destination D^i
- Paths $\mathcal{P}^i$: (sub)set of paths joining $O^i \rightsquigarrow D^i$
- Routing flow f_p: traffic along $p \in \mathcal{P} \equiv \bigcup^i \mathcal{P}^i$ generated by O/D pair owning p
- Load $x_e = \sum_{p \ni e} f_p$: total traffic along edge e
- Edge cost function $c_e(x_e)$: latency along edge e when edge load is x_e

The model



- Network: multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- O/D pairs $i \in \mathcal{I}$: origin O^i sends m^i units of traffic to destination D^i
- Paths $\mathcal{P}^i$: (sub)set of paths joining $O^i \rightsquigarrow D^i$
- Routing flow f_p: traffic along $p \in \mathcal{P} \equiv \bigcup^i \mathcal{P}^i$ generated by O/D pair owning p
- Load $x_e = \sum_{p \ni e} f_p$: total traffic along edge e
- Edge cost function $c_e(x_e)$: latency along edge e when edge load is x_e
- Path cost: $c_p(f) = \sum_{e \in p} c_e(x_e)$

The model



- Network: multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- O/D pairs $i \in \mathcal{I}$: origin O^i sends m^i units of traffic to destination D^i
- Paths $\mathcal{P}^i$: (sub)set of paths joining $O^i \rightsquigarrow D^i$
- Routing flow f_p: traffic along $p \in \mathcal{P} \equiv \bigcup^i \mathcal{P}^i$ generated by O/D pair owning p
- Load $x_e = \sum_{p \ni e} f_p$: total traffic along edge e
- Edge cost function $c_e(x_e)$: latency along edge e when edge load is x_e
- Path cost: $c_p(f) = \sum_{e \in p} c_e(x_e)$
- Nonatomic routing game: $\Gamma = (\mathcal{G}, \mathcal{I}, \{m^i\}_{i \in \mathcal{I}}, \{\mathcal{P}^i\}_{i \in \mathcal{I}}, \{c_e\}_{e \in \mathcal{E}})$

Fair/Envy-free traffic assignment

Wardrop's routing principle (Wardrop, 1952):

> *"At equilibrium, the delays along all utilized paths are equal and no higher than those that would be experienced by a traffic element going through an unused route"*

Fairness: *all traffic elements experience the same latency*

Fair/Envy-free traffic assignment

Wardrop's routing principle (Wardrop, 1952):

> *"At equilibrium, the delays along all utilized paths are equal and no higher than those that would be experienced by a traffic element going through an unused route"*

Fairness: *all traffic elements experience the same latency*

The flow profile $f^* \in \mathcal{F}$ is a *Wardrop equilibrium* if

$$c_{p^i}(f^*) \leq c_{q^i}(f^*) \quad \text{for all } i \in \mathcal{I} \text{ and all } p^i, q^i \in \mathcal{P}^i \text{ such that } f^*_{p^i} > 0$$

Fair/Envy-free traffic assignment

Wardrop's routing principle (Wardrop, 1952):

> *"At equilibrium, the delays along all utilized paths are equal and no higher than those that would be experienced by a traffic element going through an unused route"*

Fairness: *all traffic elements experience the same latency*

The flow profile $f^* \in \mathcal{F}$ is a *Wardrop equilibrium* if

$$c_{p^i}(f^*) \leq c_{q^i}(f^*) \quad \text{for all } i \in \mathcal{I} \text{ and all } p^i, q^i \in \mathcal{P}^i \text{ such that } f^*_{p^i} > 0$$

Characterization of Wardrop equilibria (Beckmann et al, 1956):

$$\begin{aligned} &\text{minimize} && \sum_{e \in \mathcal{E}} C_e(x_e) \\ &\text{subject to} && x_e = \sum_{p \ni e} f_p, \; f \in \mathcal{F} \end{aligned} \tag{WE}$$

where $C_e(x_e) = \int_0^{x_e} c_e(w)\, dw$ is the *primitive* of c_e.

Optimum traffic assignment

Efficient flows minimize aggregate latency in the network

$$\begin{aligned} &\text{minimize} && C(f) = \sum_{p \in \mathcal{P}} f_p c_p(f) \\ &\text{subject to} && f \in \mathcal{F} \end{aligned} \tag{LM}$$

The price of anarchy

- Efficient routing: $\mathrm{Opt}(\Gamma) = \min_{f \in \mathcal{F}} C(f)$
- Equilibrium routing: $\mathrm{Eq}(\Gamma) = C(f^*)$, with f^* a Wardrop equilibrium

The price of anarchy

- Efficient routing: $\mathrm{Opt}(\Gamma) = \min_{f \in \mathcal{F}} C(f)$
- Equilibrium routing: $\mathrm{Eq}(\Gamma) = C(f^*)$, with f^* a Wardrop equilibrium

Gap in efficiency measured by the *price of anarchy* (Koutsoupias & Papadimitriou, 1999)

$$\mathrm{PoA}(\Gamma) = \frac{\mathrm{Eq}(\Gamma)}{\mathrm{Opt}(\Gamma)}$$

$\mathrm{PoA}(\Gamma) \geq 1$ with equality **iff fair routing is also efficient**

How bad is selfish routing?

Theorem (Roughgarden & Tardos, 2002)

Affine cost functions ($c_e(x_e) = a_e + b_e x_e$): $\mathrm{PoA}(\Gamma) \leq 4/3$

How bad is selfish routing?

Theorem (Roughgarden & Tardos, 2002)

Affine cost functions ($c_e(x_e) = a_e + b_e x_e$): $\quad \mathrm{PoA}(\Gamma) \leq 4/3$

Theorem (Roughgarden, 2003)

Quartic (BPR) cost functions: $\quad \mathrm{PoA}(\Gamma) \leq 5\sqrt[4]{5}/(5\sqrt[4]{5} - 4) \approx 2.1505$

How bad is selfish routing?

Theorem (Roughgarden & Tardos, 2002)

Affine cost functions ($c_e(x_e) = a_e + b_e x_e$): $\quad \mathrm{PoA}(\Gamma) \leq 4/3$

Theorem (Roughgarden, 2003)

Quartic (BPR) cost functions: $\quad \mathrm{PoA}(\Gamma) \leq 5\sqrt[4]{5}/(5\sqrt[4]{5} - 4) \approx 2.1505$

Theorem (Roughgarden, 2003)

Polynomials of degree at most d: $\quad \mathrm{PoA}(\Gamma) = \mathcal{O}(d/\log d)$

How bad is selfish routing?

Theorem (Roughgarden & Tardos, 2002)
Affine cost functions ($c_e(x_e) = a_e + b_e x_e$): $\mathrm{PoA}(\Gamma) \leq 4/3$

Theorem (Roughgarden, 2003)
Quartic (BPR) cost functions: $\mathrm{PoA}(\Gamma) \leq 5\sqrt[4]{5}/(5\sqrt[4]{5}-4) \approx 2.1505$

Theorem (Roughgarden, 2003)
Polynomials of degree at most d: $\mathrm{PoA}(\Gamma) = \mathcal{O}(d/\log d)$

Remarks

- Independent of network topology
- Valid for any number of O/D pairs
- Envy-free routing can become **arbitrarily bad:** $d/\log d \to \infty$ as $d \to \infty$
- Sharpness: for any traffic inflow $M = \sum^i m^i$, these bounds can be realized

Outline

Background

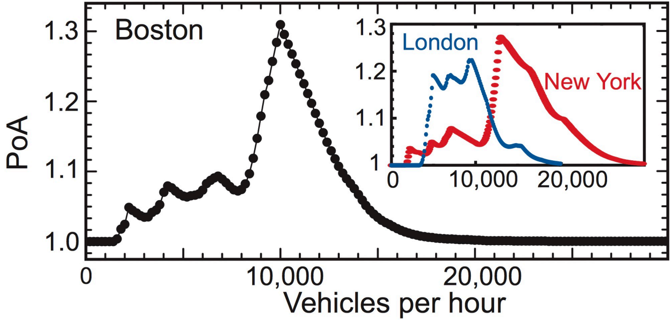
Model and preliminaries

The PoA in practice

Algorithms and dynamics

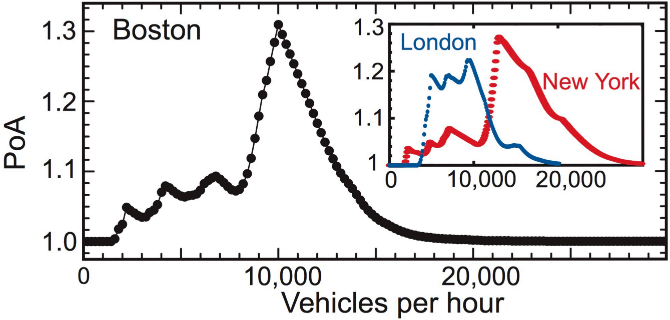
How bad is selfish routing in practice?

- PoA bounds $\Longrightarrow$ delicately tuned worst-case instances
- In typical networks, $\mathrm{PoA} \approx 1$ when the traffic is light or heavy (Youn et al., 2008; O'Hare et al., 2016; Monnot et al., 2017)



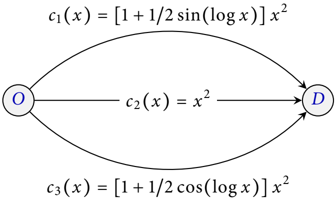
How bad is selfish routing in practice?

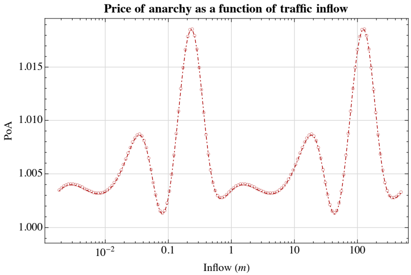
- PoA bounds $\Longrightarrow$ delicately tuned worst-case instances
- In typical networks, $\mathrm{PoA} \approx 1$ when the traffic is light or heavy (Youn et al., 2008; O'Hare et al., 2016; Monnot et al., 2017)



Is this always the case?

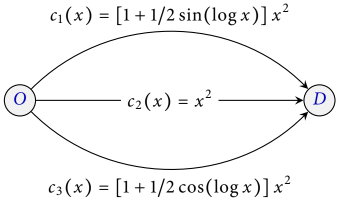
No!

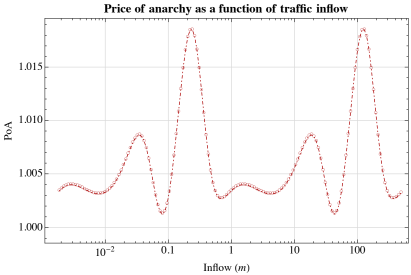




- *Single O/D pair*
- *Three parallel links (no Braess-type shenanigans)*
- C^∞*-smooth, convex cost functions with polynomial growth*

No!





- *Single O/D pair*
- *Three parallel links (no Braess-type shenanigans)*
- C^∞*-smooth, convex cost functions with polynomial growth*

Proposition (Colini-Baldeschi, Cominetti, M & Scarsini, 2017)

In the above network, $\mathrm{PoA}(\Gamma_M) \geq a > 1$ *for all values of the traffic inflow* M

What's wrong with this simple example?

Pathological oscillations

Main problem: Cost functions scale very irregularly (albeit polynomially!):

$$\lim_{t\to\{0,\infty\}} \frac{c_e(tx)}{c_e(t)} \quad \text{does not exist}$$

$\Longrightarrow$ Oscillations that are very **dense** (in light traffic) or very **wide** (in heavy traffic)

Pathological oscillations

Main problem: Cost functions scale very irregularly (albeit polynomially!):

$$\lim_{t\to\{0,\infty\}} \frac{c_e(tx)}{c_e(t)} \quad \text{does not exist}$$

$\Longrightarrow$ Oscillations that are very **dense** (in light traffic) or very **wide** (in heavy traffic)

Sanity check: *such oscillations are not observed in practice*

Regular variation

Definition (Karamata, 1930's)

A function $g\colon [0,\infty) \to (0,\infty)$ is called *regularly varying at $\omega \in \{0,\infty\}$* if

$$\lim_{t\to\omega} \frac{g(tx)}{g(t)} \quad \text{is finite and nonzero for all } x \geq 0. \tag{RV}$$

- $\omega = 0$: *light traffic limit*
- $\omega = \infty$: *heavy traffic limit*

Examples

1. Affine functions: $g(x) = ax + b$
2. Polynomials: $g(x) = \sum_{k=1}^{d} a_k x^k$
3. Asymptotic polynomials: $g(x) \sim x^q$ for some $q \geq 0$
4. Real-analytic at ω; logarithms; etc.

NB: Stronger than asking $g(x) = \Theta(x^q)$ (counterexample satisfies this)

Benchmark functions

Main idea: use a regularly varying function $c(x)$ as a *benchmark:*

Benchmark functions

Main idea: use a regularly varying function $c(x)$ as a *benchmark:*

- *Edge index:* $\alpha_e = \lim_{x \to \omega} c_e(x)/c(x)$
- *Fast / slow / tight edge:* $\alpha_e = 0, \infty$ or in-between

Benchmark functions

Main idea: use a regularly varying function $c(x)$ as a *benchmark:*

- *Edge index:* $\alpha_e = \lim_{x\to\omega} c_e(x)/c(x)$
- *Fast / slow / tight edge:* $\alpha_e = 0, \infty$ or in-between
- *Path index:* $\alpha_p = \max_{e\in p} \alpha_e$ (bottleneck caused by *slowest* edge)
- *Fast / slow / tight path:* $\alpha_p = 0, \infty$ or in-between

Benchmark functions

Main idea: use a regularly varying function $c(x)$ as a *benchmark:*

- *Edge index:* $\alpha_e = \lim_{x\to\omega} c_e(x)/c(x)$
- *Fast / slow / tight edge:* $\alpha_e = 0, \infty$ or in-between
- *Path index:* $\alpha_p = \max_{e\in p} \alpha_e$ (bottleneck caused by *slowest* edge)
- *Fast / slow / tight path:* $\alpha_p = 0, \infty$ or in-between
- *Pair index:* $\alpha^i = \min_{p\in\mathcal{P}^i} \alpha_p$ (traffic routed via *fastest* path)
- *Fast / slow / tight pair:* $\alpha^i = 0, \infty$ or in-between

Benchmark functions

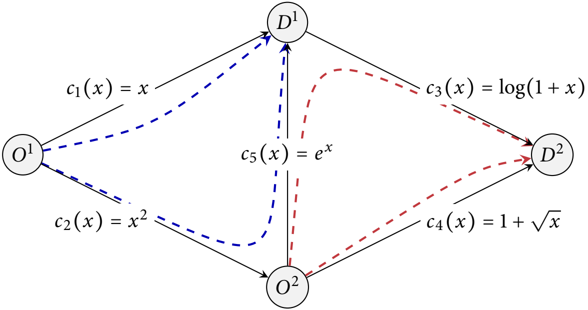
Main idea: use a regularly varying function $c(x)$ as a *benchmark:*

- *Edge index:* $\alpha_e = \lim_{x\to\omega} c_e(x)/c(x)$
- *Fast / slow / tight edge:* $\alpha_e = 0, \infty$ or in-between
- *Path index:* $\alpha_p = \max_{e\in p} \alpha_e$ (bottleneck caused by *slowest* edge)
- *Fast / slow / tight path:* $\alpha_p = 0, \infty$ or in-between
- *Pair index:* $\alpha^i = \min_{p\in\mathcal{P}^i} \alpha_p$ (traffic routed via *fastest* path)
- *Fast / slow / tight pair:* $\alpha^i = 0, \infty$ or in-between
- *Network index:* $\alpha = \min_{p\in\mathcal{P}} \alpha_p$ (bottleneck caused by *slowest* pair)
- *Tight network:* $\alpha \in (0, \infty)$

NB: Edges/paths/pairs that are slow in heavy traffic can be fast in light traffic and vice versa

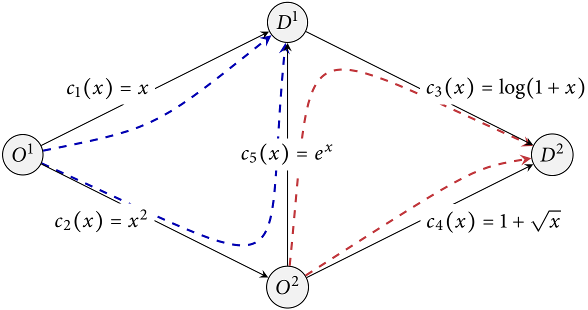
Benchmarks, light and heavy

Example: different benchmarks in a Wheatstone network



Benchmarks, light and heavy

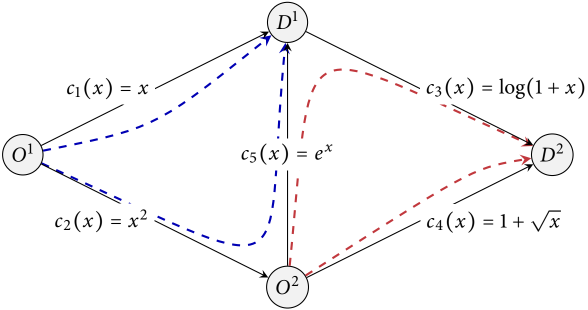
Example: different benchmarks in a Wheatstone network



- Heavy traffic, benchmark $c(x) = x$: network is tight ✓

Benchmarks, light and heavy

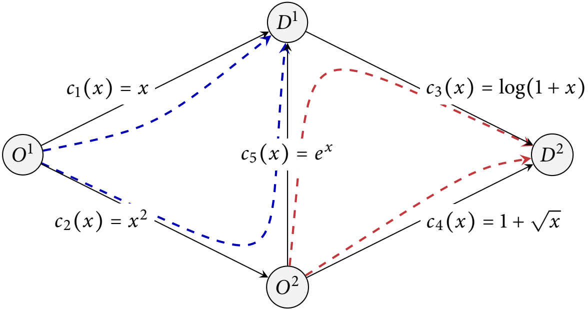
Example: different benchmarks in a Wheatstone network



- Heavy traffic, benchmark $c(x) = x$: network is tight ✓
- Light traffic, benchmark $c(x) = x$: pair 2 is slow, network not tight ✗

Benchmarks, light and heavy

Example: different benchmarks in a Wheatstone network



- Heavy traffic, benchmark $c(x) = x$: network is tight ✓
- Light traffic, benchmark $c(x) = x$: pair 2 is slow, network not tight ✗
- Light traffic, benchmark $c(x) = 1$: network is tight ✓

Asymptotic analysis

Theorem (Colini-Baldeschi, Cominetti, M & Scarsini, 2018)

Let Γ_M be a network with total traffic inflow M. If the network is tight in light $(\omega = 0)$ and/or heavy $(\omega = \infty)$ traffic, then

$$\lim_{M\to\omega} \mathrm{PoA}(\Gamma_M) = 1$$

Asymptotic analysis

Theorem (Colini-Baldeschi, Cominetti, M & Scarsini, 2018)

Let Γ_M be a network with total traffic inflow M. If the network is tight in light $(\omega = 0)$ and/or heavy $(\omega = \infty)$ traffic, then

$$\lim_{M\to\omega} \operatorname{PoA}(\Gamma_M) = 1$$

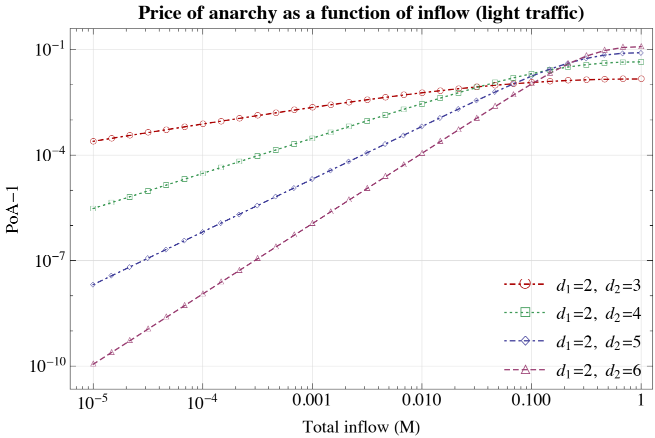
Corollary

If the network's cost functions are polynomials, $\operatorname{PoA}(\Gamma_M) \to 1$ as $M \to \omega \in \{0, \infty\}$.

In networks with polynomial costs, the gap between fairness and efficiency disappears under both light and heavy traffic

Convergence rate

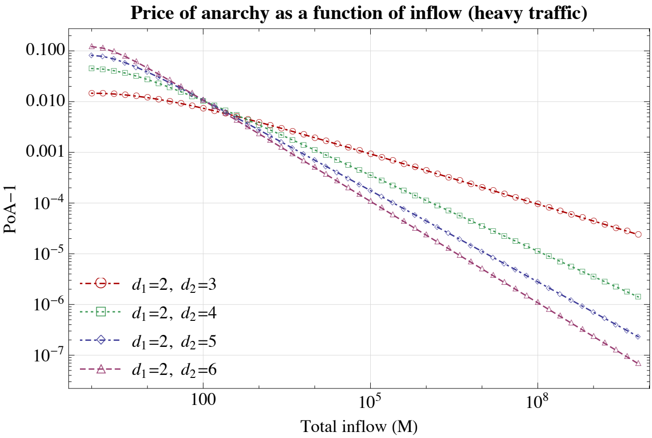
Two-link Pigou network with cost functions $c_1(x_1) = x_1^{d_1}$, $c_2(x_2) = x_2^{d_2}$



The PoA converges to 1 following a power law

Convergence rate

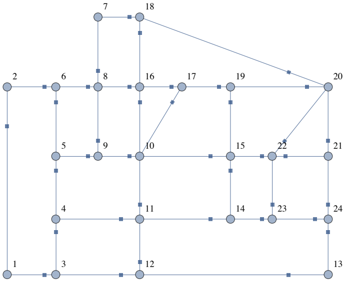
Two-link Pigou network with cost functions $c_1(x_1) = x_1^{d_1}$, $c_2(x_2) = x_2^{d_2}$



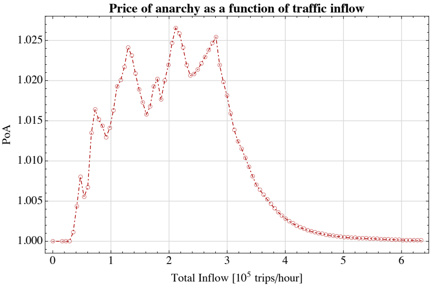
The PoA converges to 1 following a power law

Sioux Falls: a case study

The price of anarchy in Sioux Falls:



(e) The Sioux Falls road network



(f) The price of anarchy in Sioux Falls.

Average traffic inflow $M_{\text{avg}} \approx 3.6 \times 10^5$ trips/hour (LeBlanc et al., 1975)

Outline

Background

Model and preliminaries

The PoA in practice

Algorithms and dynamics

Towards optimality / equilibrium

Two (related) optimization problems:

1. Total latency minimization:

$$\begin{aligned} \text{minimize} \quad & C(f) = \sum_{e \in \mathcal{E}} x_e c_e(x_e) \\ \text{subject to} \quad & x_e = \sum_{p \ni e} f_p, \; f \in \mathcal{F} \end{aligned} \tag{LM}$$

2. Fairness (Wardrop equilibrium):

$$\begin{aligned} \text{minimize} \quad & L(f) = \sum_{e \in \mathcal{E}} C_e(x_e) \\ \text{subject to} \quad & x_e = \sum_{p \ni e} f_p, \; f \in \mathcal{F} \end{aligned} \tag{WE}$$

where $C_e(x_e) = \int_0^{x_e} c_e(w)\,dw$ is the *primitive* of c_e

How can either problem be solved in a scalable and efficient manner?

Challenges

Generic problem formulation (single O/D pair for simplicity):

$$\begin{aligned} \text{minimize} \quad & g(f) = \sum_{e \in \mathcal{E}} g_e(x_e) \\ \text{subject to} \quad & f \in \mathcal{F} \end{aligned} \qquad \text{(Opt)}$$

where:

- $x_e = \sum_{p \ni e} f_p$ [edge-route duality]
- $\mathcal{F} = M \cdot \Delta(\mathcal{P}) = \{f \in \mathbb{R}_+^{\mathcal{P}} : \sum_{p \in \mathcal{P}} f_p = M\}$ [simplicial structure]

Challenges

Generic problem formulation (single O/D pair for simplicity):

$$\begin{aligned} \text{minimize} \quad & g(f) = \sum_{e \in \mathcal{E}} g_e(x_e) \\ \text{subject to} \quad & f \in \mathcal{F} \end{aligned} \tag{Opt}$$

where:

- $x_e = \sum_{p \ni e} f_p$ [edge-route duality]
- $\mathcal{F} = M \cdot \Delta(\mathcal{P}) = \{f \in \mathbb{R}_+^{\mathcal{P}} : \sum_{p \in \mathcal{P}} f_p = M\}$ [simplicial structure]

Challenges:

- Information: cost functions a priori unknown
- Dimensionality: exponential number of paths
- Control plane: dynamic/distributed flow control

Flow/load gradients

Generic problem formulation:

$$\begin{aligned} \text{minimize} \quad & g(f) = \sum_{e \in \mathcal{E}} g_e(x_e) \\ \text{subject to} \quad & f \in \mathcal{F} \end{aligned} \tag{Opt}$$

Flow/load gradients

Generic problem formulation:

$$\begin{aligned} \text{minimize} \quad & g(f) = \sum_{e \in \mathcal{E}} g_e(x_e) \\ \text{subject to} \quad & f \in \mathcal{F} \end{aligned} \tag{Opt}$$

Follow the negative gradient of g:

$$v = -\nabla g$$

By edge-route duality:

$$v_p(f) = -\frac{\partial g}{\partial f_p} = -\sum_{e \in \mathcal{E}} g_e'(x_e) \frac{\partial x_e}{\partial f_p} = -\sum_{e \in p} g_e'(x_e) =: \sum_{e \in p} v_e(x_e)$$

To get route flow gradient ← sum edge load gradients along route

Gradient information

Assume algorithmic scheme generates at $n = 1, 2, \ldots$

- Flow profile $f_n = (f_{p,n})_{p \in \mathcal{P}}$
- Load profile $x_n = (x_{e,n})_{e \in \mathcal{E}}$, $x_{e,n} = \sum_{e \in p} f_{p,n}$

Leverage gradient information to update, but cost function g_e a priori unknown

Gradient information

Assume algorithmic scheme generates at $n = 1, 2, \ldots$

- Flow profile $f_n = (f_{p,n})_{p \in \mathcal{P}}$
- Load profile $x_n = (x_{e,n})_{e \in \mathcal{E}}$, $x_{e,n} = \sum_{e \in p} f_{p,n}$

Leverage gradient information to update, but cost function g_e a priori unknown

When called at x_n, $n = 1, 2, \ldots$, assume gradients estimated up to some (random) error:

$$\hat{v}_{e,n} = v_e(x_{e,n}) + U_{e,n+1}$$

with the following hypotheses for the *error process* U:

(H1) **Zero-mean:** $\mathbb{E}[U_{n+1} \,|\, \mathcal{F}_n] = 0$

(H2) **Finite variance:** $\mathbb{E}[\|U_{n+1}\|^2 \,|\, \mathcal{F}_n] \leq \sigma^2$

Gradient descent

Projected gradient descent:

$$f_{n+1} = \Pi_{\mathcal{F}}(f_n + \gamma_n \hat{v}_n) \tag{GD}$$

where

$$\Pi_{\mathcal{F}}(f) = \arg\min_{f' \in \mathcal{F}} \| f' - f \|$$

is the Euclidean projection on $\mathcal{F}$ (simplicial projection)

Gradient descent

Projected gradient descent:

$$f_{n+1} = \Pi_{\mathcal{F}}(f_n + \gamma_n \hat{v}_n) \tag{GD}$$

where

$$\Pi_{\mathcal{F}}(f) = \arg\min_{f' \in \mathcal{F}} \| f' - f \|$$

is the Euclidean projection on $\mathcal{F}$ (simplicial projection)

Theorem (folk)

Suppose that $|g_e'| \leq G$. *If* (GD) *is run with* $\gamma_n \propto 1/\sqrt{Gn}$ *and returns*

$$\bar{f}_n = \frac{\sum_{k=1}^n \gamma_k f_k}{\sum_{k=1}^n \gamma_k},$$

then

$$\mathbb{E}[g(\bar{f}_n)] \leq \min g + \mathcal{O}(G\sqrt{|\mathcal{P}|/n})$$

The good

Theorem (folk)

Suppose that $|g_e'| \leq G$. *If* (GD) *is run with* $\gamma_n \propto 1/\sqrt{Gn}$ *and returns*

$$\bar{f}_n = \frac{\sum_{k=1}^{n} \gamma_k f_k}{\sum_{k=1}^{n} \gamma_k},$$

then

$$\mathbb{E}[g(\bar{f}_n)] \leq \min g + \mathcal{O}(G\sqrt{|\mathcal{P}|/n})$$

- Value convergence despite imperfect feedback ✓
- Can be improved to convergence in high probability ✓

The good, the (not so) bad

Theorem (folk)

Suppose that $|g'_e| \leq G$*. If* (GD) *is run with* $\gamma_n \propto 1/\sqrt{Gn}$ *and returns*

$$\bar{f}_n = \frac{\sum_{k=1}^{n} \gamma_k f_k}{\sum_{k=1}^{n} \gamma_k},$$

then

$$\mathbb{E}[g(\bar{f}_n)] \leq \min g + \mathcal{O}(G\sqrt{|\mathcal{P}|/n})$$

- Value convergence despite imperfect feedback ✓
- Can be improved to convergence in high probability ✓
- Rate in n cannot be improved (but not too slow in practice) ✓

The good, the (not so) bad, and the ugly

Theorem (folk)

Suppose that $|g_e'| \leq G$. *If* (GD) *is run with* $\gamma_n \propto 1/\sqrt{Gn}$ *and returns*

$$\bar{f}_n = \frac{\sum_{k=1}^n \gamma_k f_k}{\sum_{k=1}^n \gamma_k},$$

then

$$\mathbb{E}[g(\bar{f}_n)] \leq \min g + \mathcal{O}(G\sqrt{|\mathcal{P}|/n})$$

- Value convergence despite imperfect feedback ✓
- Can be improved to convergence in high probability ✓
- Rate in n cannot be improved (but not too slow in practice) ✓
- **Exponential dependence on the size of the graph because of $|\mathcal{P}|$** ✗
- **Projection step has complexity $\Theta(|\mathcal{P}| \log|\mathcal{P}|)$** ✗
- **Need to store $\Theta(|\mathcal{P}|)$ variables** ✗

Exponentiated gradient descent

An idea from **reinforcement learning** (Vovk, Littlestone & Warmuth, ...):

- Keep a score for each path, based on its performance so far
- Allocate traffic proportionally to the exponential of this score

Exponentiated gradient descent

An idea from **reinforcement learning** (Vovk, Littlestone & Warmuth, ...):

- Keep a score for each path, based on its performance so far
- Allocate traffic proportionally to the exponential of this score

Key insight: score by aggregating (negative) gradient steps

Exponentiated gradient descent (EGD)

Require: step-size sequence $\gamma_k > 0$

1: set $y_p \leftarrow 0$ for each route $p \in \mathcal{P}$ # initialization
2: **for** $k = 1, 2, \dots, n$ **do**
3: assign traffic $f_p \propto \exp(y_p)$ # exponential weights
4: set $y \leftarrow y + \gamma_k \hat{v}_k$ # score update
5: **end for**
6: **return** $\bar{f}_n = \sum_{k=1}^n \gamma_k f_k \Big/ \sum_{k=1}^n \gamma_k$

The good

Theorem (M, Paschos, Vigneri, 2018)

Suppose that $|g'_e| \leq G$. *If EGD is run with* $\gamma_n \propto 1/\sqrt{Gn}$, *then*

$$\mathbb{E}[g(f_n)] \leq \min g + \mathcal{O}(G\sqrt{\log|\mathcal{P}|/n})$$

- **Linear dependence on the size of the graph through $|\mathcal{P}|$** ✓
- Value convergence despite imperfect feedback ✓
- Can be improved to convergence in high probability ✓

The good, the (not so) bad

Theorem (M, Paschos, Vigneri, 2018)

Suppose that $|g'_e| \leq G$. *If EGD is run with* $\gamma_n \propto 1/\sqrt{Gn}$, *then*

$$\mathbb{E}[g(f_n)] \leq \min g + \mathcal{O}(G\sqrt{\log|\mathcal{P}|/n})$$

- **Linear dependence on the size of the graph through $|\mathcal{P}|$** ✓
- Value convergence despite imperfect feedback ✓
- Can be improved to convergence in high probability ✓
- Rate in n cannot be improved (but not too slow in practice) ✓

The good, the (not so) bad, and the ugly

Theorem (M, Paschos, Vigneri, 2018)

Suppose that $|g_e'| \leq G$. *If EGD is run with* $\gamma_n \propto 1/\sqrt{Gn}$, *then*

$$\mathbb{E}[g(f_n)] \leq \min g + \mathcal{O}(G\sqrt{\log|\mathcal{P}|/n})$$

- **Linear dependence on the size of the graph through $|\mathcal{P}|$** ✓
- Value convergence despite imperfect feedback ✓
- Can be improved to convergence in high probability ✓
- Rate in n cannot be improved (but not too slow in practice) ✓
- **Normalization step has complexity $\Theta(|\mathcal{P}|)$** ✗
- **Still need to store $\Theta(|\mathcal{P}|)$ variables** ✗

Distribution in the control plane

Can we distribute the algorithm at the node level?

- Given: an O/D pair (O, D)
- Each node $v \in \mathcal{V}$ has a subset of edges e_v that can be used to reach D
- No backtracking: acyclic routing (multi-)graph $\mathcal{G} = (\mathcal{V}, \bigcup_{v \in \mathcal{V}} e_v)$
- Each node controls traffic allocation over $\mathcal{E}_v$, i.e., a vector

$$z = (z_e)_{e \in \mathcal{E}_v} \in \Delta(\mathcal{E}_v)$$

- Small dimensionality per control node – **but how to implement EGD?**

The role of weight propagation

Key steps in EGD:

- Update scores: $y_e \leftarrow y_e + \gamma \hat{v}_e$ ✓
- Traffic allocation: ??? ✗

Straightforward choice of weights:

$$z_e = \frac{\exp(y_e)}{\sum_{e' \in \mathcal{E}_v} \exp(y_{e'})}$$

OK in terms of dimension; **complete failure in terms of optimization**

Backpedaling

Key insight: must not be blind to what is happening down the road!

0. **Require:** edge score vector $y = (y_e)_{e \in \mathcal{E}}$

   Initialize: latent weight variables w_v for each $v \in \mathcal{V}$, w_e for each $e \in \mathcal{E}$.
   Set $w_D = 0$ at destination; backpropagate w_D through all edges linking to D.

Backpedaling

Key insight: must not be blind to what is happening down the road!

0. **Require:** edge score vector $y = (y_e)_{e \in \mathcal{E}}$

   Initialize: latent weight variables w_v for each $v \in \mathcal{V}$, w_e for each $e \in \mathcal{E}$.
   Set $w_D = 0$ at destination; backpropagate w_D through all edges linking to D.

1. **Weigh and wait:** When node v receives weight information from connecting node v' via edge $e \in \mathcal{E}_v$, set

$$w_e = y_e + w_{v'}$$

Backpedaling

Key insight: must not be blind to what is happening down the road!

0. **Require:** edge score vector $y = (y_e)_{e \in \mathcal{E}}$

   Initialize: latent weight variables w_v for each $v \in \mathcal{V}$, w_e for each $e \in \mathcal{E}$.
   Set $w_D = 0$ at destination; backpropagate w_D through all edges linking to D.

1. **Weigh and wait:** When node v receives weight information from connecting node v' via edge $e \in \mathcal{E}_v$, set

$$w_e = y_e + w_{v'}$$

2. **Sum and send:** If node v has received an update via all outgoing edges $\mathcal{E}_v$, set

$$w_v = \log \sum_{e \in \mathcal{E}_v} \exp(w_e)$$

   and push w_v back through all edges linking to v

Exponential weights and backpedaling

Proposition

Let $y \in \mathbb{R}^{\mathcal{E}}$ be an edge score vector and suppose each node $v \in \mathcal{V}$ allocates traffic following the exponential rule

$$z_e = \frac{\exp(w_e)}{\exp(w_v)} \quad \text{for all } e \in \mathcal{E}_v,$$

with w_e and w_v defined via backpedaling. Then, the total traffic flowing through route $p \in \mathcal{P}$ is

$$f_p = \frac{\exp(y_p)}{\sum_{q \in \mathcal{P}} \exp(y_q)}$$

where $y_p = \sum_{e \in p} y_e$ denotes the corresponding path score.

Exponential node weights with backpedaling induce exponential path weights!

Distributed EGD

Theorem (Gaujal, Héliou, M, 2018)

Suppose that $|g'_e| \leq G$. *If EGD is run at the node level with backpedaling and a step-size* $\gamma_n \propto 1/\sqrt{Gn}$, *then*

$$\mathbb{E}[g(\bar{f}_n)] \leq \min g + \mathcal{O}(G\sqrt{\log|\mathcal{P}|/n})$$

- Value convergence despite imperfect feedback ✓
- Can be improved to convergence in high probability ✓

Distributed EGD

Theorem (Gaujal, Hélious, M, 2018)

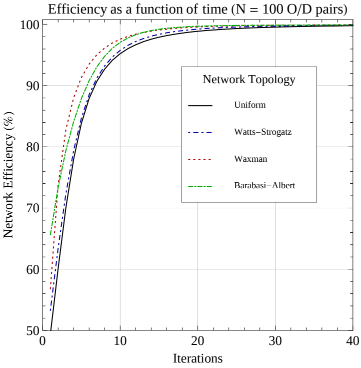
Suppose that $|g_e'| \leq G$. *If EGD is run at the node level with backpedaling and a step-size* $\gamma_n \propto 1/\sqrt{Gn}$, *then*

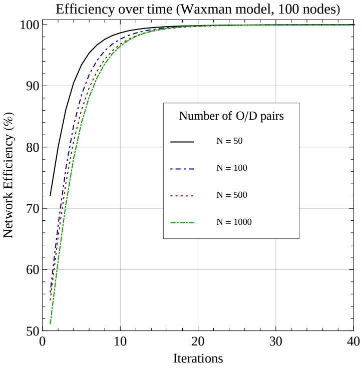
$$\mathbb{E}[g(\bar{f}_n)] \leq \min g + \mathcal{O}(G\sqrt{\log|\mathcal{P}|/n})$$

- Value convergence despite imperfect feedback ✓
- Can be improved to convergence in high probability ✓
- **Linear dependence on the size of the graph through $|\mathcal{P}|$** ✓
- **Update step has $\mathcal{O}(|\mathcal{E}_v|)$ complexity per node** ✓
- **Only need to store $\mathcal{O}(|\mathcal{E}_v|)$ variables per node** ✓

Convergence rate

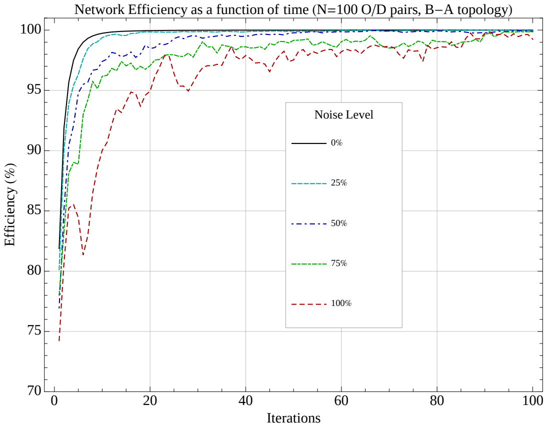
Distributed EGD in randomly generated networks





Convergence rate

Distributed EGD in randomly generated networks



Conclusions and perspectives

"Traffic congestion is caused by vehicles, not by people in themselves"

— Jane Jacobs, *The Death and Life of Great American Cities*

- The price of anarchy disappears in light and heavy traffic, independently of the network topology and even with multiple O/D pairs
- Exponential weights + backpedaling allow fast, distributed optimization
- **Size of the network:** not a curse, but a blessing in disguise

Conclusions and perspectives

"Traffic congestion is caused by vehicles, not by people in themselves"

— Jane Jacobs, *The Death and Life of Great American Cities*

- The price of anarchy disappears in light and heavy traffic, independently of the network topology and even with multiple O/D pairs
- Exponential weights + backpedaling allow fast, distributed optimization
- **Size of the network:** not a curse, but a blessing in disguise

Open questions

- Capacitated networks (M/M/1, M/G/1, etc.)?
- What if there is **no** gradient feedback whatsoever?
- What about **atomic** routing games?