ΔΡΟΜΟΛΟΓΗΣΗ ΥΠΟ ΣΥΝΘΗΚΕΣ ΑΒΕΒΑΙΟΤΗΤΑΣ ΣΕ ΔΙΚΤΥΑ ΜΕΓΑΛΗΣ ΚΛΙΜΑΚΑΣ

Παναγιώτης Μερτικόπουλος

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Τμήμα Μαθηματικών

⟨ Σεμινάριο Στατιστικής & Επιχ. Έρευνας | ΕΚΠΑ, Τμήμα Μαθηματικών | 1 Μαρτίου, 2023 ⟩

Outline

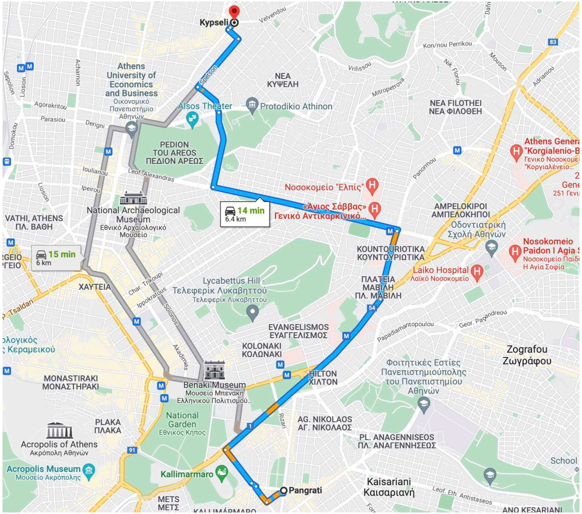
Traffic...

...how bad can it get?

Traffic...

...how bad can it get?

Game of roads



Athens at a glance

- 3, 754, 000 people
- 937, 000 daily trips
- Up to 10^4 trips/min
- 1393 nodes
- 5429 edges
- 1, 360, 000 O/D pairs
- $\approx 7 * 10^{18}$ paths

A very large game!

Two overarching questions

Part 1: ***How bad is selfish routing, really?***

- The price of anarchy: worst-case bounds and beyond
- When practice meets theory

Part 2: ***How to reach an equilibrium?***

- Optimal algorithms: from uncertainty to acceleration
- Universal algorithms: optimal rates without prior knowledge

The people


K. Antonakopoulos


R. Colini-Baldeschi


R. Cominetti


Y. G. Hsieh

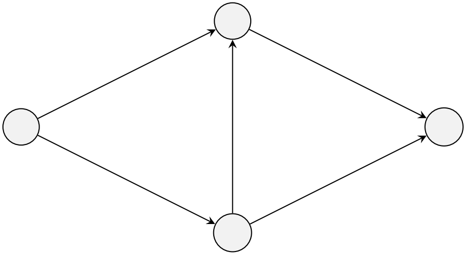

M. Scarsini


D. Q. Vu

- Antonakopoulos & **M.**, *Adaptive first-order methods revisited: Convex optimization without Lipschitz requirements*. NeurIPS 2021
- Antonakopoulos, Vu, Cevher, Levy & **M.**, *UnderGrad: A universal black-box optimization method with almost dimension-free convergence rate guarantees*. ICML 2022
- Colini-Baldeschi, Cominetti, **M.** & Scarsini, *The asymptotic behavior of the price of anarchy*. WINE 2017
- Colini-Baldeschi, Cominetti, **M.** & Scarsini, *When is selfish routing bad? The price of anarchy in light and heavy traffic*. Operations Research, vol. 68(2), pp. 411-434, 2020.
- Hsieh, Antonakopoulos & **M.**, *Adaptive learning in continuous games: Optimal regret bounds and convergence to Nash equilibrium*. COLT 2021
- Vu, Antonakopoulos & **M.**, *Fast routing under uncertainty: Adaptive learning in congestion games with exponential weights*. NeurIPS 2021

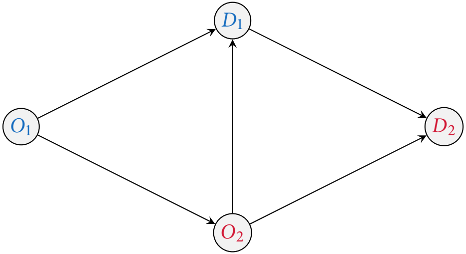
Outline

Nonatomic congestion games



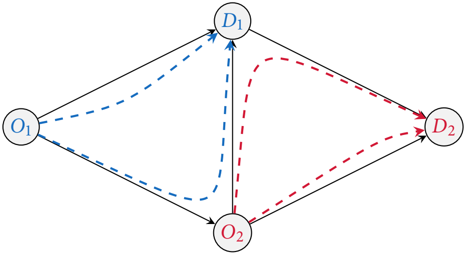
- ***Network:*** multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

Nonatomic congestion games



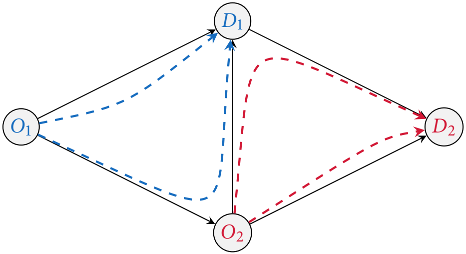
- ***Network:*** multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- ***O/D pairs $i \in \mathcal{N}$:*** origin O_i sends m_i units of traffic to destination D_i

Nonatomic congestion games



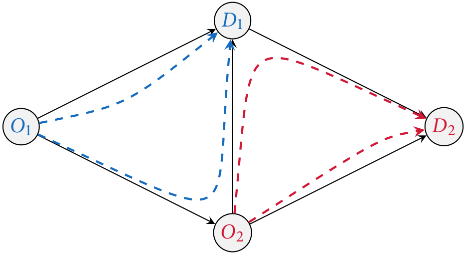
- ***Network:*** multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- ***O/D pairs $i \in \mathcal{N}$:*** origin O_i sends m_i units of traffic to destination D_i
- ***Paths $\mathcal{P}_i$:*** (sub)set of paths joining $O_i \rightsquigarrow D_i$

Nonatomic congestion games



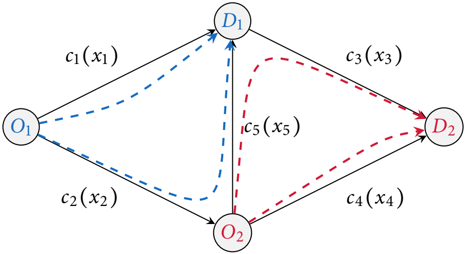
- ***Network:*** multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- ***O/D pairs*** $i \in \mathcal{N}$***:*** origin O_i sends m_i units of traffic to destination D_i
- ***Paths*** $\mathcal{P}_i$***:*** (sub)set of paths joining $O_i \rightsquigarrow D_i$
- ***Routing flow*** f_p***:*** traffic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

Nonatomic congestion games



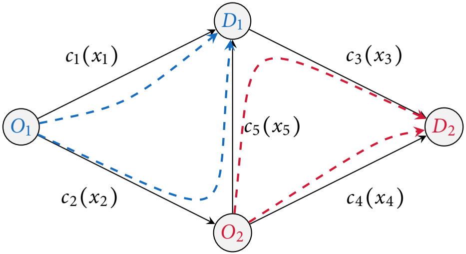
- ***Network:*** multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- ***O/D pairs $i \in \mathcal{N}$:*** origin O_i sends m_i units of traffic to destination D_i
- ***Paths $\mathcal{P}_i$:*** (sub)set of paths joining $O_i \rightsquigarrow D_i$
- ***Routing flow f_p:*** traffic along $p \in \mathcal{P}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

Nonatomic congestion games



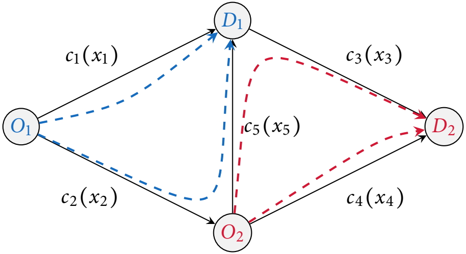
- ***Network:*** multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- ***O/D pairs $i \in \mathcal{N}$:*** origin O_i sends m_i units of traffic to destination D_i
- ***Paths $\mathcal{P}_i$:*** (sub)set of paths joining $O_i \rightsquigarrow D_i$
- ***Routing flow f_p:*** traffic along $p \in \mathcal{P}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

Nonatomic congestion games



- ***Network:*** multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- ***O/D pairs*** $i \in \mathcal{N}$**:** 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*** $c_e(x_e)$**:** cost along edge e when edge load is x_e
- ***Path cost:*** $c_p(f) = \sum_{e \in p} c_e(x_e)$

Nonatomic congestion games



- ***Network:*** multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- ***O/D pairs*** $i \in \mathcal{N}$**:** 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*** $c_e(x_e)$**:** cost along edge e when edge load is x_e
- ***Path cost:*** $c_p(f) = \sum_{e \in p} c_e(x_e)$
- **Nonatomic congestion game:** $\mathcal{G} = (\mathcal{G}, \mathcal{N}, \{m_i\}_{i \in \mathcal{N}}, \{\mathcal{P}_i\}_{i \in \mathcal{N}}, \{c_e\}_{e \in \mathcal{E}})$

Traffic equilibrium

Wardrop equilibrium

A flow profile $f^* \in \mathcal{F} \equiv \{f \in \mathbb{R}_+^{\mathcal{P}} : \sum_{p \in \mathcal{P}_i} f_p = m_i\}$ is a ***Wardrop equilibrium*** if

$$c_{p_i}(f^*) \leq c_{q_i}(f^*) \quad \text{for all utilized paths } p_i \in \mathcal{P}_i, i \in \mathcal{N} \tag{WE}$$

Equilibrium routing is envy-free: all traffic elements experience the same latency

Theorem (Beckmann et al., 1956)

A flow profile f^* is a Wardrop equilibrium if and only if it solves the convex problem

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in \mathcal{E}} \int_0^{x_e} c_e(w)\, dw \\ \text{subject to} \quad & x_e = \sum_{p \ni e} f_p, \ f \in \mathcal{F} \end{aligned} \tag{Eq}$$

Price of Anarchy

Optimal flows

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

Price of Anarchy (Koutsoupias & Papadimitriou, 1999; Papadimitriou, 2001)

Equilibrium cost: $\mathrm{Eq}(\mathcal{G}) = C(f^*)$

Minimum cost: $\mathrm{Opt}(\mathcal{G}) = \min_{f \in \mathcal{F}} C(f)$

Price of Anarchy: $\mathrm{PoA}(\mathcal{G}) = \dfrac{\mathrm{Eq}(\mathcal{G})}{\mathrm{Opt}(\mathcal{G})}$

How bad is selfish routing?

Theorem (Roughgarden & Tardos, 2002; Roughgarden, 2003)

- *Affine cost functions* ($c_e(x_e) = a_e + b_e x_e$)

$$\mathrm{PoA}(\mathcal{G}) \leq 4/3$$

- *Quartic (BPR) cost functions*

$$\mathrm{PoA}(\mathcal{G}) \leq 5\sqrt[4]{5}/(5\sqrt[4]{5} - 4) \approx 2.1505$$

- *Polynomials of degree at most d*

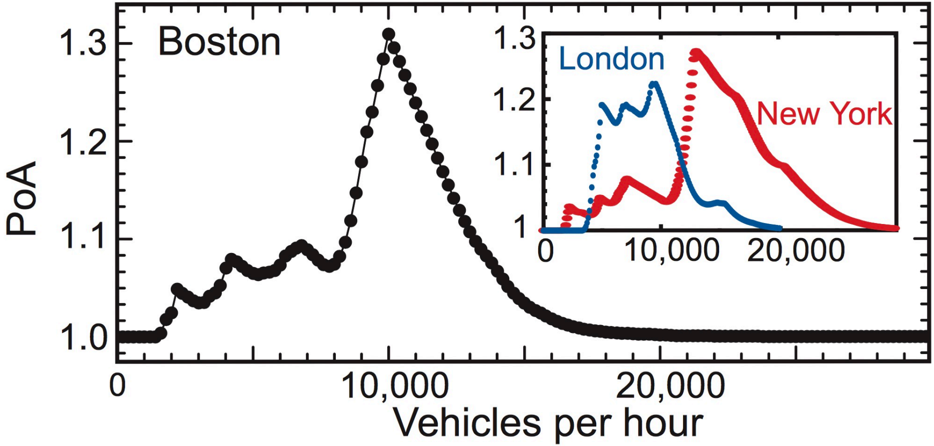
$$\mathrm{PoA}(\mathcal{G}) = \mathcal{O}(d/\log d)$$

Remarks

- Independent of network topology
- Valid for any number of O/D pairs
- Equilibrium routing can become **arbitrarily bad:** $d/\log d \to \infty$ as $d \to \infty$

How bad is selfish routing, really?

Delicately tuned worst-case instances are not representative of reality



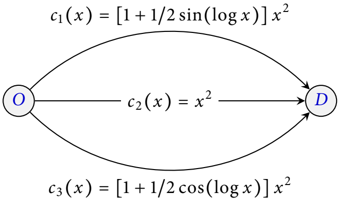
Source: Youn et al., 2008

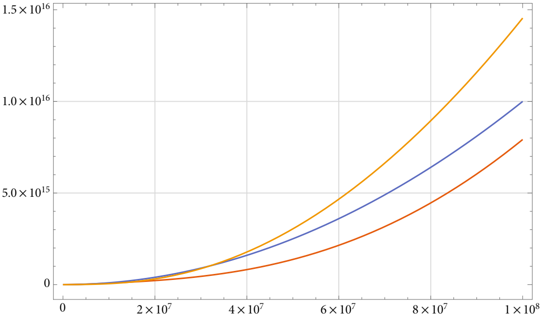
Price of anarchy: asymptotics

Does the price of anarchy always vanish in the limit?

Price of anarchy: asymptotics

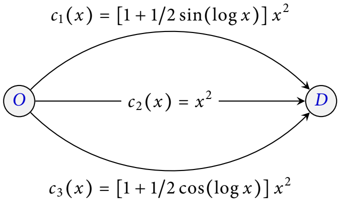
Does the price of anarchy always vanish in the limit?

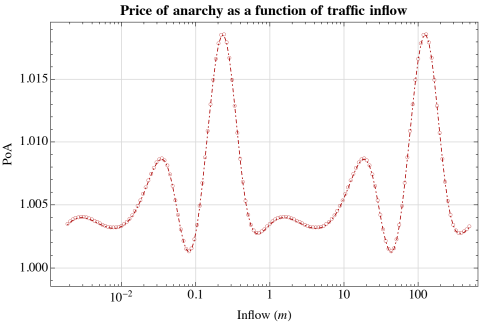




Price of anarchy: asymptotics

Does the price of anarchy always vanish in the limit?





Proposition (Colini-Baldeschi, Cominetti, M & Scarsini, 2020)

In the above network:

$$\inf_M \mathrm{PoA}(\mathcal{G}_M) > 1$$

Pathological oscillations

Cost functions are C^∞-smooth, convex and grow polynomially - **but irregularly:**

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

- ***In light traffic:*** infinitely **dense** oscillations
- ***In heavy traffic:*** infinitely **wide** oscillations
- **Sanity check:** no such oscillations observed in practice

Regular variation

Definition (Karamata, 1930's)

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

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

- ***Light traffic:*** $\omega = 0$
- ***Heavy traffic:*** $\omega = \infty$

Examples

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

NB: $\Theta(x^q) \not\subseteq \text{(RV)} \not\subseteq \Theta(x^q)$

Network benchmarks

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

Network benchmarks

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

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

Network benchmarks

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

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

Network benchmarks

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

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

Network benchmarks

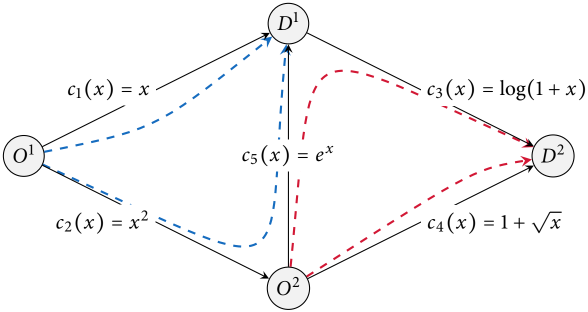
Main idea: find a regularly varying function $c(x)$ to use as a benchmark:

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

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

Benchmarks, light and heavy

Example: light and heavy traffic benchmarks in a Wheatstone network



- ***Heavy traffic benchmark:*** $c(x) = x$
- ***Light traffic benchmark:*** $c(x) = 1$

The price of anarchy in light and heavy traffic

Theorem (Colini-Baldeschi, Cominetti, M & Scarsini, 2020)

Assume: *the network admits a regularly varying benchmark function*

Then: $\mathrm{PoA}(\mathcal{G}_M) \to 1$ *as* $M \to \{0, \infty\}$

The price of anarchy in light and heavy traffic

Theorem (Colini-Baldeschi, Cominetti, M & Scarsini, 2020)

Assume: *the network admits a regularly varying benchmark function*

Then: $\mathrm{PoA}(\mathcal{G}_M) \to 1$ *as* $M \to \{0, \infty\}$

Corollary

In networks with polynomial cost functions, $\mathrm{PoA}(\mathcal{G}_M) \to 1$ *as* $M \to \{0, \infty\}$.

Outline

The road to equilibrium

How to reach an equilibrium state?

- Lack of information — # Will it rain in the next hour?
- Very large problems — # $\approx 10^8$ user base

The road to equilibrium

How to reach an equilibrium state?

- Lack of information — # Will it rain in the next hour?
- Very large problems — # $\approx 10^8$ user base

Recommender must be able to solve in real time:

$$\begin{aligned} \text{minimize} \quad & L(f) = \sum_{e \in \mathcal{E}} \int_0^{x_e} c_e(w)\, dw \\ \text{subject to} \quad & x_e = \sum_{p \ni e} f_p, \; f \in \mathcal{F} \end{aligned} \tag{WE}$$

Challenges

- **Variability:** traffic conditions fluctuate unpredictably
- **Uncertainty:** congestion metrics only partially observable
- **Dimensionality:** exponential number of state variables

The model

Randomness and uncertainty:

- ***Exogenous randomness*** $\omega \in \Omega$ reflected in observed costs $\rightsquigarrow c_e(x_e; \omega)$

  # "State of the world": weather, accidents, added congestion...

- ***Mean equilibrium flows***

$$\mathbb{E}_\omega[c_{p_i}(f^*; \omega)] \leq \mathbb{E}_\omega[c_{q_i}(f^*; \omega)] \quad \text{for all utilized paths } p_i \in \mathcal{P}_i, i \in \mathcal{N}$$

Sequence of events

1: **for all** $t = 1, 2, \ldots$ **do**
2: Interface recommends flow profile $f_t \in \mathcal{F}$
3: Nature determines state of the network $\omega_t \in \Omega$
4: Traffic elements on path p incur $c_p(f_t; \omega_t)$
5: **end for**

Equilibrium characterization

Stochastic convex programming characterization

f^* is a ***mean equilibrium flow*** if and only if it solves

$$\begin{aligned} \text{minimize} \quad & \bar{L}(f) = \mathbb{E}\left[\sum_{e \in \mathcal{E}} \int_0^{x_e} c_e(u;\omega)\, du\right] \\ \text{subject to} \quad & x_e = \sum_{p \ni e} f_p, \; f \in \mathcal{F} \end{aligned} \tag{$\overline{\text{Eq}}$}$$

NB: *Observed cost vectors* $\rightsquigarrow$ *stochastic gradients*

$$\nabla \bar{L}(f) = (\bar{c}_p(f))_{p \in \mathcal{P}} = \mathbb{E}\left[(c_p(f;\omega))_{p \in \mathcal{P}}\right]$$

Two sharply different regimes:

- *Static:* ω_t remains constant with time
- *Stochastic:* ω_t fluctuates with time

Stochastic gradient descent

Stochastic gradient descent:

$$f_{t+1} = \mathrm{pr}_{\mathcal{F}}(f_t - \gamma \hat{c}_t) \tag{SGD}$$

where $\hat{c}_t = c(f_t; \omega_t)$ is the ***cost profile*** at time t and $\gamma > 0$ is a ***step-size*** parameter

Stochastic gradient descent

Stochastic gradient descent:

$$f_{t+1} = \mathrm{pr}_{\mathcal{F}}(f_t - \gamma \hat{c}_t) \tag{SGD}$$

where $\hat{c}_t = c(f_t; \omega_t)$ is the ***cost profile*** at time t and $\gamma > 0$ is a ***step-size*** parameter

Theorem (folk)

If (SGD) is run for T iterations with $\gamma \propto 1/\sqrt{T}$, the mean flow $\bar{f}_T = T^{-1} \sum_{t=1}^{T} f_t$ enjoys

$$\mathbb{E}[\bar{L}(\bar{f}_T) - \min \bar{L}] = \mathcal{O}\left(\sqrt{P/T}\right)$$

Stochastic gradient descent

Stochastic gradient descent:

$$f_{t+1} = \mathrm{pr}_{\mathcal{F}}(f_t - \gamma \hat{c}_t) \tag{SGD}$$

where $\hat{c}_t = c(f_t; \omega_t)$ is the ***cost profile*** at time t and $\gamma > 0$ is a ***step-size*** parameter

Theorem (folk)

If (SGD) is run for T iterations with $\gamma \propto 1/\sqrt{T}$, the mean flow $\bar{f}_T = T^{-1} \sum_{t=1}^{T} f_t$ enjoys

$$\mathbb{E}[\bar{L}(\bar{f}_T) - \min \bar{L}] = \mathcal{O}\left(\sqrt{P/T}\right)$$

Properties:

- ✓ **Optimal in T:** query complexity cannot be improved in the **stochastic** regime
- ✗ **Slow in P:** query complexity is **exponential** in the network's size
- ✗ **Non-adaptive:** requires tuning of γ
- ✗ **Offline:** $\bar{f}_t$ is never recommended

Routing with exponential weights

The **exponential weights (ExpWeight)** algorithm # mirror descent for the simplex

$$f_{p,t+1} \propto f_{p,t} \exp(-\gamma \hat{c}_{p,t}) \tag{EW}$$

where "∝" indicates normalization over all paths belonging to the same O/D pair

Routing with exponential weights

The **exponential weights (ExpWeight)** algorithm # mirror descent for the simplex

$$f_{p,t+1} \propto f_{p,t} \exp(-\gamma \hat{c}_{p,t}) \tag{EW}$$

where "$\propto$" indicates normalization over all paths belonging to the same O/D pair

Theorem (Blum et al., 2006)

If ExpWeight *is run for T steps with $\gamma \propto 1/\sqrt{T}$, the mean flow $\bar{f}_T = T^{-1} \sum_{t=1}^{T} f_t$ enjoys*

$$\bar{L}(\bar{f}_T) - \min \bar{L} = \mathcal{O}\left(\sqrt{\log P / T}\right)$$

Routing with exponential weights

The **exponential weights (ExpWeight)** algorithm # mirror descent for the simplex

$$f_{p,t+1} \propto f_{p,t} \, \exp(-\gamma \hat{c}_{p,t}) \tag{EW}$$

where "$\propto$" indicates normalization over all paths belonging to the same O/D pair

Theorem (Blum et al., 2006)

If ExpWeight *is run for T steps with $\gamma \propto 1/\sqrt{T}$, the mean flow $\bar{f}_T = T^{-1} \sum_{t=1}^{T} f_t$ enjoys*

$$\bar{L}(\bar{f}_T) - \min \bar{L} = \mathcal{O}\left(\sqrt{\log P / T}\right)$$

Properties:

- ✓ **Optimal in T:** query complexity cannot be improved in the **stochastic** regime
- ✓ **Optimal in P:** query complexity is **polynomial** in the network's size
- ✗ **Non-adaptive:** requires tuning of γ
- ✗ **Offline:** $\bar{f}_t$ is never recommended

The static case

Is the situation the same in static the static regime?

- ✓ Nesterov's accelerated gradient (NAG) method achieves $\mathcal{O}(1/T^2)$ in static programs
- ✗ **But exponential dependence on $|\mathcal{G}|$**

Can we get rates that are optimal in both T and P?

The static case

Is the situation the same in static the static regime?

- ✓ Nesterov's accelerated gradient (NAG) method achieves $\mathcal{O}(1/T^2)$ in static programs
- ✗ **But exponential dependence on $|\mathcal{G}|$**

> Can we get rates that are optimal in both T and P?

Algorithm Accelerated exponential weights (AcceleWeight) — # NAG + ExpWeight

Require: initial score vector $y_0 \leftarrow 0$; moving weight $\alpha_0 \leftarrow 0$; step $\gamma_0 \leftarrow 1/(NM\beta)$ — # $\beta \rightsquigarrow$ Lipschitz modulus

```
for all t = 1, 2, ... T do
    set z_t ∝ exp(y_{t-1})                                          # ExpWeight step
    set f_t ← α_{t-1} f_{t-1} + (1 − α_{t-1}) z_t                   # Nesterov momentum
    set γ_t ← ½[2γ_{t-1} + γ_0 + sqrt(4γ_{t-1}γ_0 + γ_0²)]           # NAG step-size
    set α_t ← γ_{t-1}/γ_t                                           # moving weight update
    set z̄_t ← α_t f_t + (1 − α_t) z_t and get c_t ← c(z̄_t)          # route and measure costs
    set y_t ← y_{t-1} − (1 − α_t) γ_t c_t                           # update path scores
end for
return f_t                                                          # output flow
```

$z_t \propto \exp(y_{t-1})$, $f_t \leftarrow \alpha_{t-1} f_{t-1} + (1-\alpha_{t-1}) z_t$, $\gamma_t \leftarrow \frac{1}{2}\left[2\gamma_{t-1} + \gamma_0 + \sqrt{4\gamma_{t-1}\gamma_0 + \gamma_0^2}\right]$, $\alpha_t \leftarrow \gamma_{t-1}/\gamma_t$, $\bar{z}_t \leftarrow \alpha_t f_t + (1-\alpha_t) z_t$, $c_t \leftarrow c(\bar{z}_t)$, $y_t \leftarrow y_{t-1} - (1-\alpha_t)\gamma_t c_t$

AcceleWeight guarantees

Theorem (Vu et al., 2021)

In the static regime, AcceleWeight *enjoys the rate of convergence*

$$L(f_T) - \min L \leq \frac{4\beta^2 N^2 M^2 \log P}{T^2} = \mathcal{O}\left(\frac{\log P}{T^2}\right)$$

AcceleWeight guarantees

Theorem (Vu et al., 2021)

In the static regime, AcceleWeight *enjoys the rate of convergence*

$$L(f_T) - \min L \leq \frac{4\beta^2 N^2 M^2 \log P}{T^2} = \mathcal{O}\left(\frac{\log P}{T^2}\right)$$

Properties:

- ✓ **Optimal in T:** query complexity cannot be improved in the **static** regime
- ✓ **Optimal in P:** query complexity is **polynomial** in the network's size
- ✗ **Non-adaptive:** requires tuning of γ
- ✗ **Offline:** f_t is never recommended

The good

The good:

- ✓ In the stochastic regime, ExpWeight is **optimal** in T and P
- ✓ In the static regime, AcceleWeight is **optimal** in T and P

The good, the bad

The good:

✓ In the stochastic regime, ExpWeight is **optimal** in T and P

✓ In the static regime, AcceleWeight is **optimal** in T and P

The bad:

✗ In the static regime, ExpWeight is **very slow** in T

✗ In the stochastic regime, AcceleWeight **does not converge**

The good, the bad, *and the ugly*

The good:

- ✓ In the stochastic regime, ExpWeight is **optimal** in T and P
- ✓ In the static regime, AcceleWeight is **optimal** in T and P

The bad:

- ✗ In the static regime, ExpWeight is **very slow** in T
- ✗ In the stochastic regime, AcceleWeight **does not converge**

The ugly:

- Tuning the step-size is impractical / impossible
- Output is never recommended

Adaptive algorithms

Observe:

- In the static regime: $\|c_{t+1} - c_t\|_\infty$ should become small over time
- In the stochastic regime: $\|c_{t+1} - c_t\|_\infty$ remains bounded away from zero

Adaptive algorithms

Observe:

- In the static regime: $\|c_{t+1} - c_t\|_\infty$ should become small over time
- In the stochastic regime: $\|c_{t+1} - c_t\|_\infty$ remains bounded away from zero

Adaptive step-size (Rakhlin & Sridharan, 2013; Hsieh, Antonakopoulos & M, 2021)

$$\gamma_t = \frac{1}{\sqrt{1 + \sum_{s=1}^{t-1} \|c_{s+1} - c_s\|_\infty^2}} \tag{Adapt}$$

Adaptive algorithms

Observe:

- In the static regime: $\|c_{t+1} - c_t\|_\infty$ should become small over time
- In the stochastic regime: $\|c_{t+1} - c_t\|_\infty$ remains bounded away from zero

Adaptive step-size (Rakhlin & Sridharan, 2013; Hsieh, Antonakopoulos & M, 2021)

$$\gamma_t = \frac{1}{\sqrt{1 + \sum_{s=1}^{t-1} \|c_{s+1} - c_s\|_\infty^2}} \tag{Adapt}$$

Algorithm ExpWeight + Adapt # Antonakopoulos & M, 2021

Initialize score vector $y \in \mathbb{R}^{\mathcal{P}}$
1: **for all** $t = 1, 2, \dots T$ **do**
2: Route according to $f_t \sim \exp(y_t)$ # ExpWeight update
3: Observe cost profile: $\hat{c}_t \leftarrow (c_p(f_t; \omega_t))_{p \in \mathcal{P}}$ # cost feedback
4: Update path scores: $y_{t+1} \leftarrow y_t - \gamma_t \hat{c}_t$ # Adapt step
5: **end for**
6: **return** $\bar{f}_T = (1/T) \sum_{t=1}^{T} f_t$ # output flow

Guarantees of ExpWeight + Adapt

Theorem (Antonakopoulos & M, 2021)

Suppose that ExpWeight +Adapt *is run for T steps. Then $\bar{f}_T$ enjoys the rate*

$$\mathbb{E}[\bar{L}(\bar{f}_T) - \min \bar{L}] = \mathcal{O}\left(\frac{\log(PT)}{T} + \sigma\sqrt{\frac{\log(PT)}{T}}\right)$$

where σ^2 is the variance of $\|c'(x;\omega)\|_{\mathcal{L}^1}$.

Properties:

- ✓ **Optimal in stochastic regime:** query complexity cannot be improved in T if $\sigma > 0$
  - **Better** than ExpWeight in the static regime, but **worse** than AcceleWeight
- ✓ **Adaptive:** no hyperparameter tuning required
- ✗ **Offline:** $\bar{f}_t$ is never recommended

AdaWeight

Is there a path to universal acceleration?

AdaWeight

Is there a path to universal acceleration?

Algorithm Adaptive exponential weights (AdaWeight) — # Vu et al., 2021

Initialize score vector $y_1 \leftarrow 0$; moving weight $\alpha_0 \leftarrow 0$; step $\eta_1 \leftarrow 1$

for all $t = 1, 2, \dots T$ **do**

- set $z_t \propto \exp(\eta_t y_t)$ — # ExpWeight step
- set $\bar{z} \leftarrow \left(\alpha_t z_t + \sum_{s=0}^{t-1} \alpha_s z_{s+1/2}\right) / \sum_{s=0}^{t} \alpha_s$ and get $\bar{c}_t \leftarrow c(\bar{z}_t; \omega_t)$ — # reweigh + explore
- set $y_{t+1/2} \leftarrow y_t - \alpha_t \bar{c}_t$ — # score update
- set $z_{t+1/2} \propto \exp(\eta_t y_{t+1/2})$ — # ExpWeight step
- set $f_t \leftarrow \left(\sum_{s=0}^{t} \alpha_s z_{s+1/2}\right) / \sum_{s=0}^{t} \alpha_s$ and get $c_t \leftarrow c(f_t; \omega_t)$ — # route and measure costs
- set $y_{t+1} \leftarrow y_t - \gamma_t c_t$ — # update scores
- set $\eta_{t+1} \leftarrow \eta_t \Big/ \sqrt{1 + \eta_t^2 \alpha_t^2 \|c_t - \bar{c}_t\|_\infty^2}$ — # Adapt step

end for

return f_t — # output flow

Borrows ideas from ExpWeight + NAG + dual extrapolation methods

AdaWeight guarantees

Theorem (Vu et al., 2021; Antonakopoulos et al., 2022)

AdaWeight *enjoys the rate of convergence*

$$\mathbb{E}[L(f_T) - \min L] = \mathcal{O}\left(\frac{\log P}{T^2} + \frac{\sigma \log P}{\sqrt{T}}\right)$$

Properties:

- ✓ **Optimal in stochastic regime:** query complexity cannot be improved in T if $\sigma > 0$
- ✓ **Optimal in static regime:** query complexity cannot be improved in T if $\sigma = 0$
- ✓ **Fast in P:** query complexity is **polynomial** in the network's size
- ✓ **Adaptive:** does not require any tuning or prior system knowledge
- ✓ **Online:** guarantees concern the recommended flows

AdaWeight in practice

Numerical experiments in the Anaheim metropolitan area

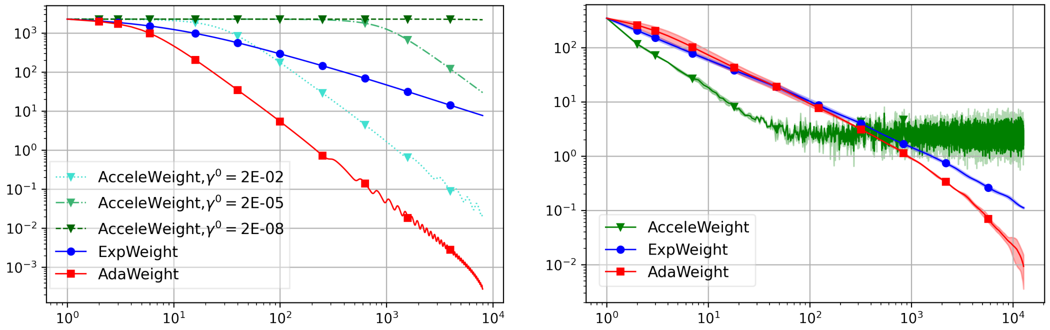


Figure: ExpWeight, AcceleWeight & AdaWeight in static (left) and stochastic (right) conditions

Two overarching questions

Q1: ***How bad is selfish routing, really?***

✓ **Not too bad:** in realistic network conditions, no difference between selfish and socially optimum states

✓ Price of anarchy **vanishes** under low and heavy traffic

Q2: ***Is it possible to reach an equilibrium efficiently?***

✓ Adaptive routing methods can achieve "best of all worlds" guarantees

- No tuning required
- Optimal in both static and stochastic regimes
- Smooth transition between static and stochastic
- Polynomial - as opposed to exponential - in network size

References I

Antonakopoulos, K. and Mertikopoulos, P. Adaptive first-order methods revisited: Convex optimization without Lipschitz requirements. In *NeurIPS '21: Proceedings of the 35th International Conference on Neural Information Processing Systems*, 2021.

Antonakopoulos, K., Vu, D. Q., Cevher, V., Levy, K. Y., and Mertikopoulos, P. UnderGrad: A universal black-box optimization method with almost dimension-free convergence rate guarantees. In *ICML '22: Proceedings of the 39th International Conference on Machine Learning*, 2022.

Beckmann, M., McGuire, C. B., and Winsten, C. *Studies in the Economics of Transportation*. Yale University Press, 1956.

Blum, A., Even-Dar, E., and Ligett, K. Routing without regret: on convergence to Nash equilibria of regret-minimizing algorithms in routing games. In *PODC '06: Proceedings of the 25th annual ACM SIGACT-SIGOPS symposium on Principles of Distributed Computing*, pp. 45–52, 2006.

Colini-Baldeschi, R., Cominetti, R., Mertikopoulos, P., and Scarsini, M. The asymptotic behavior of the price of anarchy. In *WINE 2017: Proceedings of the 13th Conference on Web and Internet Economics*, 2017.

Colini-Baldeschi, R., Cominetti, R., Mertikopoulos, P., and Scarsini, M. When is selfish routing bad? The price of anarchy in light and heavy traffic. *Operations Research*, 68(2):411–434, March 2020.

Hsieh, Y.-G., Antonakopoulos, K., and Mertikopoulos, P. Adaptive learning in continuous games: Optimal regret bounds and convergence to Nash equilibrium. In *COLT '21: Proceedings of the 34th Annual Conference on Learning Theory*, 2021.

Koutsoupias, E. and Papadimitriou, C. H. Worst-case equilibria. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pp. 404–413, 1999.

Papadimitriou, C. H. Algorithms, games, and the Internet. In *STOC '01: Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, 2001.

Rakhlin, A. and Sridharan, K. Optimization, learning, and games with predictable sequences. In *NIPS '13: Proceedings of the 27th International Conference on Neural Information Processing Systems*, 2013.

References II

Roughgarden, T. The price of anarchy is independent of the network topology. *Journal of Computer and System Sciences*, 67(2):341-364, 2003.

Roughgarden, T. and Tardos, É. How bad is selfish routing? *Journal of the ACM*, 49(2):236-259, 2002.

Vu, D. Q., Antonakopoulos, K., and Mertikopoulos, P. Fast routing under uncertainty: Adaptive learning in congestion games with exponential weights. In *NeurIPS '21: Proceedings of the 35th International Conference on Neural Information Processing Systems*, 2021.

Youn, H., Gastner, M. T., and Jeong, H. Price of anarchy in transportation networks: Efficiency and optimality control. *Physical Review Letters*, 101 (12):128701, September 2008.

Outline

UnderGrad: The theory under the hood

Is there a path to universal acceleration ***for arbitrary domains?***

UnderGrad: The theory under the hood

Is there a path to universal acceleration ***for arbitrary domains?***

Dual extrapolation (DE)

$$\begin{aligned} y_{t+1/2} &= y_t - \gamma_t g_t & f_{t+1/2} &= Q(\eta_t y_{t+1/2}) \\ y_{t+1} &= y_t - \gamma_t g_{t+1/2} & f_{t+1} &= Q(\eta_{t+1} y_{t+1}) \end{aligned} \tag{DE}$$

UnderGrad: The theory under the hood

Is there a path to universal acceleration ***for arbitrary domains?***

Dual extrapolation (DE)

$$\begin{aligned} y_{t+1/2} &= y_t - \gamma_t g_t & \quad f_{t+1/2} &= Q(\eta_t y_{t+1/2}) \\ y_{t+1} &= y_t - \gamma_t g_{t+1/2} & \quad f_{t+1} &= Q(\eta_{t+1} y_{t+1}) \end{aligned} \tag{DE}$$

Adaptive learning rate

$$\eta_{t+1} = \sqrt{\frac{K_h (R_h + K_h \|\mathcal{X}\|^2)}{K_h + \sum_{s=1}^{t} \gamma_s^2 \|g_{s+1/2} - g_s\|^2}} \tag{Adapt}$$

UnderGrad: The theory under the hood

Is there a path to universal acceleration ***for arbitrary domains?***

Dual extrapolation (DE)

$$
\begin{aligned}
y_{t+1/2} &= y_t - \gamma_t g_t & f_{t+1/2} &= Q(\eta_t y_{t+1/2}) \\
y_{t+1} &= y_t - \gamma_t g_{t+1/2} & f_{t+1} &= Q(\eta_{t+1} y_{t+1})
\end{aligned}
\tag{DE}
$$

Adaptive learning rate

$$
\eta_{t+1} = \sqrt{\frac{K_h(R_h + K_h\|\mathcal{X}\|^2)}{K_h + \sum_{s=1}^{t} \gamma_s^2 \|g_{s+1/2} - g_s\|^2}}
\tag{Adapt}
$$

Iterate averaging

$$
\bar{f}_t = \frac{\gamma_t f_t + \sum_{s=1}^{t-1} \gamma_s f_{s+1/2}}{\sum_{s=1}^{t} \gamma_s}
$$

$$
\bar{f}_{t+1/2} = \frac{\gamma_t f_{t+1/2} + \sum_{s=1}^{t-1} \gamma_s f_{s+1/2}}{\sum_{s=1}^{t} \gamma_s}
$$

UnderGrad: The theory under the hood

Is there a path to universal acceleration ***for arbitrary domains?***

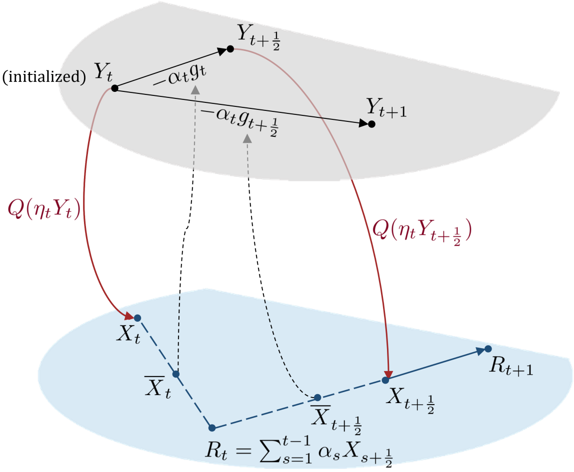


Figure: The UnderGrad algorithm

UnderGrad: The theory under the hood

Is there a path to universal acceleration ***for arbitrary domains?***

Theorem (Antonakopoulos et al., 2022)

Suppose that UnderGrad *is run for T iterations with $\gamma_t = t$. Then the algorithm's output state $\bar{x}_T \equiv \bar{f}_{T+1/2}$ concurrently enjoys the following guarantees:*

a) *If f satisfies (LC)/(BG), then*

$$\mathbb{E}[f(\bar{x}_T) - \min f] \leq 2C_h \sqrt{\frac{K_h + 8(G^2 + \sigma^2)}{K_h T}}$$

b) *If f satisfies (LS)/(LG), then*

$$\mathbb{E}[f(\bar{x}_T) - \min f] \leq \frac{32\sqrt{2}C_h^2 L}{K_h T^2} + \frac{8\sqrt{2}C_h \sigma}{\sqrt{K_h T}}$$

where $C_h = \sqrt{R_h + K_h \|\mathcal{X}\|^2}$.

Outline

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

$$\chi = (\chi_e)_{e\in\mathcal{E}_v} \in \Delta(\mathcal{E}_v)$$

- 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:

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

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*

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

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!