# Collective Dynamics for Electrical Flow Estimation

Vincenzo Bonifaci

Istituto di Analisi dei Sistemi ed Informatica (IASI-CNR)
Consiglio Nazionale delle Ricerche, Italy

joint work with
L. Becchetti (Sapienza U. Rome), E. Natale (MPII Saarbrücken)

CAALM, Chennai Mathematical Institute
21–25 January 2019

# Combinatorial network optimization

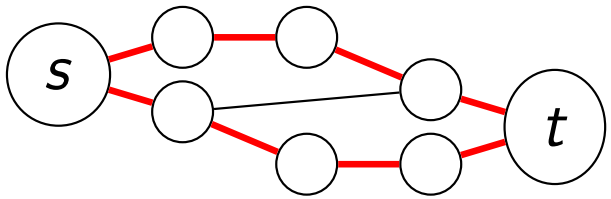
Fundamental examples of network optimization problems:

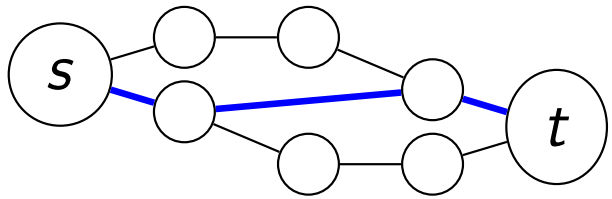
**Maximum Flow**

Find a maximum number of edge-disjoint $s$-$t$ paths

**Shortest Path**

Find an $s$-$t$ path of minimum length




Classic algorithms for Maximum Flow and Shortest Path are combinatorial: manipulate discrete objects (nodes, edges, paths...)

Computational complexity expressed in terms of:

- $n$: number of nodes
- $m$: number of edges

# Hybrid combinatorial-numerical methods

Since 2004, a new generation of fast algorithms is emerging:

Reduce network problems to solving equations of the form

$$\boxed{L\,x = b}$$

where $L \in \mathbb{R}^{n \times n}$ is a graph Laplacian matrix

**Theorem (Spielman-Teng 2004 and subsequent work)**

*A Laplacian linear system can be solved up to error $\epsilon$ in time*

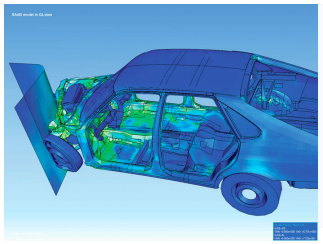
$$O\left(m \cdot \log n \cdot \log \frac{1}{\epsilon}\right) = \tilde{O}(m)$$

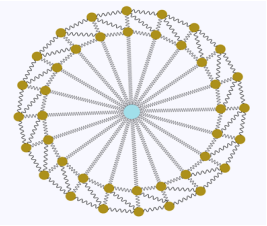
"Laplacian paradigm": build around this algorithmic primitive

# The Laplacian Paradigm

**Directly related**:

Elliptic systems



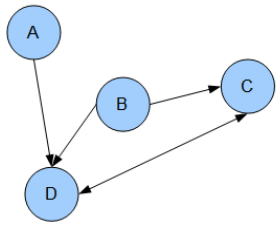


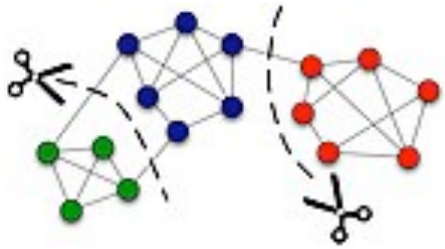
Lx=b

**Few iterations**:

Eigenvectors,

Heat kernels

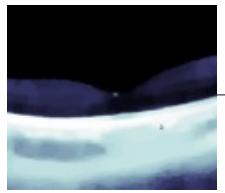


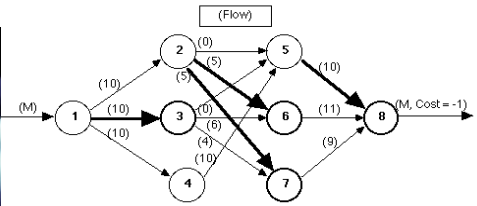



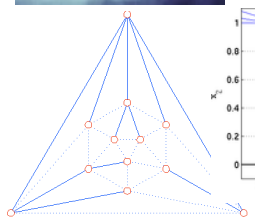
**Many iterations / modify algorithm**

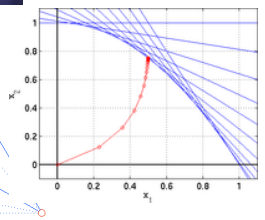
Graph problems

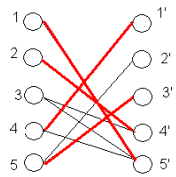
Image processing










Slide by Richard Peng

# Electrical flows as a network primitive

A representative example of a Laplacian system:

Computing currents and voltages in a resistive electrical network

A crucial subroutine in many fast network algorithms:

- Maximum flows (Christiano et al. STOC 2010)
- Shortest paths with negative weights (Cohen et al. SODA 2017)
- Network sparsification (Spielman and Srivastava 2011)

Also the basis of some models of biological computation:

- *Physarum polycephalum* slime mold (B. et al. SODA 2012)
- Ant colonies (Ma et al. 2013)

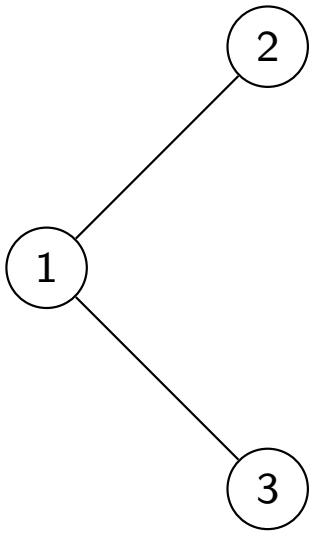
1. Laplacian framework
2. Electrical flows
3. Decentralized solution of $Lp = b$
   - Jacobi's method
   - Token diffusion

1. Laplacian framework
2. Electrical flows
3. Decentralized solution of $Lp = b$
   - Jacobi's method
   - Token diffusion

# The Laplacian matrix

$x_{uv}$: weight of an edge $u \sim v$
$d_u$: total weight of the edges around $u$ (volume or gen. degree)

$$L_{u,v} = \begin{cases} d_u & \text{if } u = v \\ -x_{uv} & \text{if } u \sim v \\ 0 & \text{otherwise.} \end{cases}$$




$$L = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} = D - A$$

$D = \mathrm{diag}(d)$

$A =$ weighted adjacency matrix

# The Laplacian matrix II

The Laplacian is positive semidefinite: $v^\top L v \geq 0$ for any $v \in \mathbb{R}^n$

$L = BXB^\top$ where:

- $B$ is the $n \times m$ incidence matrix, e.g.:

$$B = \overbrace{\left.\begin{pmatrix} +1 & +1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix}\right\}}^{\text{edges}} \text{nodes}$$

- $X$ is a diagonal $m \times m$ weight matrix, e.g.:

$$X = \left.\begin{pmatrix} x_{1,2} & 0 \\ 0 & x_{1,3} \end{pmatrix}\right\} \text{edges}$$

# Normalized Laplacian

The normalized Laplacian is

$$\mathcal{L} = D^{-1/2} L D^{-1/2}$$

where

$$D = \begin{pmatrix} d_1 & \dots & 0 \\ 0 & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & d_n \end{pmatrix}$$

For $d$-regular graphs, $\mathcal{L} = L/d$

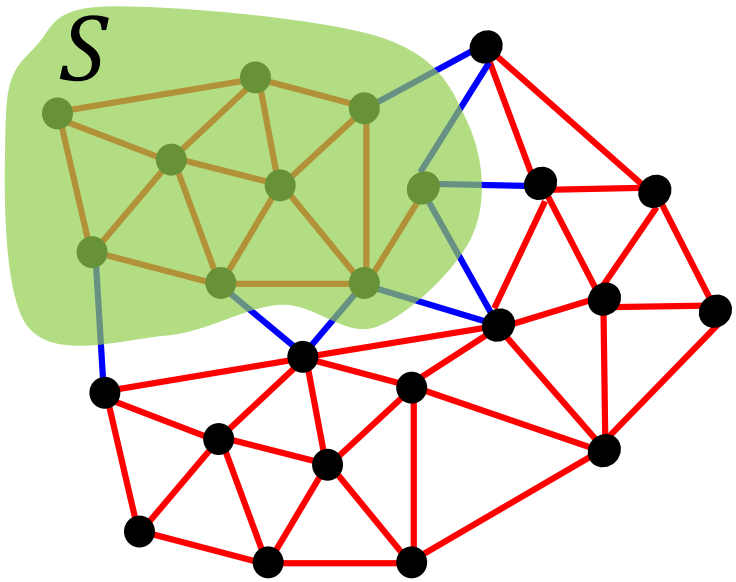
The eigenvalues of $\mathcal{L}$ satisfy

$$\begin{aligned} 0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n \leq 2 \\ \lambda_2 > 0 \Leftrightarrow \text{ the network is connected} \end{aligned} \tag{1}$$

# Cuts and conductance

A cut is a bipartition of the nodes into two sets $(S, N \setminus S)$

The weight of a cut is the total weight of edges with one endpoint in $S$ and one in $N \setminus S$: $\sum_{u \in S, v \in N \setminus S} x_{uv}$




Conductance of $S$:

$$\phi(S) = \frac{x(S, N \setminus S)}{d(S)}$$

# Cuts with small conductance

The conductance of a graph is

$$\phi_G = \min_{d(S)\leq d(N)/2} \phi(S)$$

## Cheeger inequality (1971; 1985)

$$\frac{\lambda_2}{2} \leq \phi_G \leq \sqrt{2\lambda_2}$$

where $\lambda_2$ is the second smallest eigenvalue of the normalized Laplacian $\mathcal{L}$

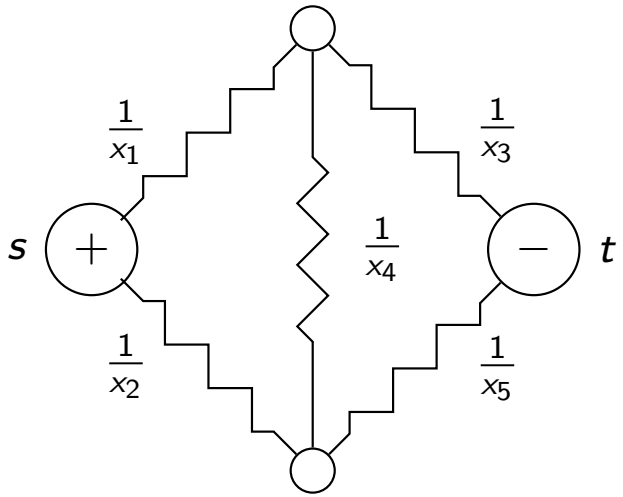
1. Laplacian framework

2. Electrical flows

3. Decentralized solution of $Lp = b$
   - Jacobi's method
   - Token diffusion

# Electrical flows

- Undirected graph $G$
- $N$: nodes, $E$: edges
- $s$, $t \in N$: source and sink of flow
- edge $e$ has conductance $x_e$
- equivalently, resistance $r_e = 1/x_e$



# Poisson's equation

The node potentials $p \in \mathbb{R}^n$ are the solutions to Poisson's equation:

$$\boxed{L \cdot p = b} \qquad \text{with (say) } b_u = \begin{cases} +1 & \text{if } u = s \\ -1 & \text{if } u = t \\ 0 & \text{otherwise} \end{cases}$$

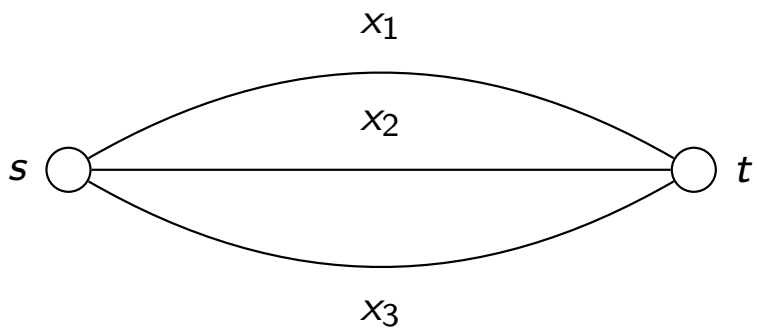
A flow is a vector $f \in \mathbb{R}^m$ that satisfies flow conservation:

$$\boxed{B \cdot f = b} \qquad (B = \text{ incidence matrix})$$

The electrical flow $q \in \mathbb{R}^m$ is related to $p$ by Ohm's law:

$$\boxed{q = X \cdot B^\top \cdot p}$$

# Example: a parallel-links network




$$B = \begin{pmatrix} +1 & +1 & +1 \\ -1 & -1 & -1 \end{pmatrix} \qquad X = \begin{pmatrix} x_1 & 0 & 0 \\ 0 & x_2 & 0 \\ 0 & 0 & x_3 \end{pmatrix}$$

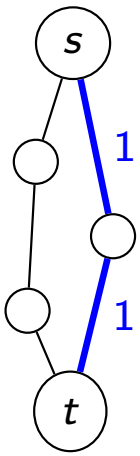
$$L(x) = BXB^\top = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \sum_{e \in E} x_e$$

$$p_s(x) = \left( \sum_{e \in E} x_e \right)^{-1} \qquad p_t(x) = 0 \qquad q_j(x) = \left( \sum_{e \in E} x_e \right)^{-1} x_j$$

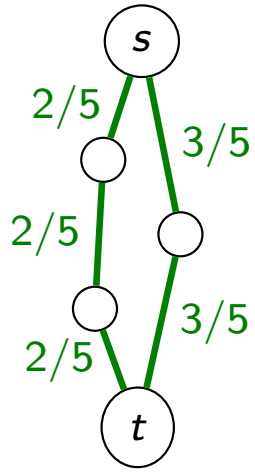
# Combinatorial flows vs. electrical flow

For unit $s$-$t$ flows $f \in \mathbb{R}^m$:

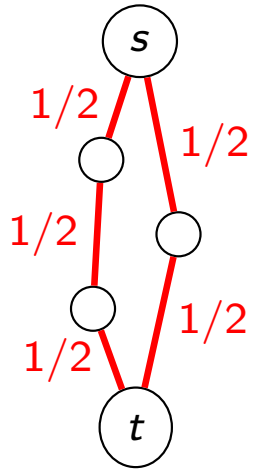
- The shortest path minimizes $\|f\|_1$
- The electrical flow minimizes $\|f\|_2$
- The (normalized) maximum flow minimizes $\|f\|_\infty$




minimize $\|f\|_1$
s.t. $Bf = b$




minimize $\|f\|_2$
s.t. $Bf = b$




minimize $\|f\|_\infty$
s.t. $Bf = b$

# From electrical flows to maximum flow

Reducing maximum flow to electrical flows (Christiano et al. 2010)
**Intuition**: increase the resistance of edges with excess flow

## Algorithm sketch

- Set resistance $r_e \leftarrow 1$ for each edge $e$
- Repeat:
  1. Laplacian solve: find the electrical flow $q$ with respect to $r$
  2. Update: increase $r_e$ proportionally to $r_e q_e$

Process converges to a maximum flow

# From electrical flows to shortest path

Reducing shortest path to electrical flows (Becchetti et al. 2013)
**Intuition**: increase the conductance of edges with excess flow

## Algorithm sketch

- Set conductance $x_e \leftarrow 1$ for each edge $e$
- Repeat:
  1. Laplacian solve: find the electrical flow $q$ with respect to $x$
  2. Update: increase $x_e$ proportionally to $q_e - x_e$

Process converges to a shortest path flow

1. Laplacian framework

2. Electrical flows

3. Decentralized solution of $Lp = b$
   - Jacobi's method
   - Token diffusion

# Decentralized solution of Laplacian systems

Consider a connected network $G$ with weights (conductances) $x \in \mathbb{R}^m$

Can we solve $L(x)\, p = b$ through a decentralized process?

We consider two approaches:

1. Jacobi's method (deterministic; send/receive real numbers)
2. Token diffusion (stochastic; send/receive tokens)

# Jacobi's method

An iterative method that can be applied to any positive-definite linear system; in our setting,

$$p_u^{(k+1)} = \frac{b_u + \sum_{v \sim u} x_{uv} p_v^{(k)}}{\sum_{v \sim u} x_{uv}}, \qquad k = 0, 1, \ldots$$

Node $u$ maintains information of $p_u^{(k)}$ and $b_u$
To update node $u$, need information only from the neighbors of $u$

It is well-known that Jacobi's method is convergent and

$$L \cdot p^{(k)} \to b \qquad \text{as } k \to \infty$$

But how fast in terms of the network parameters?

# Convergence rate of Jacobi's method

## Theorem 1

The error in Jacobi's method converges to zero at rate

$$O(\max(|1-\lambda_2|^k, |1-\lambda_n|^k))$$

where $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \leq 2$ are the eigenvalues of the normalized Laplacian $\mathcal{L}$ of the network.

**Corollary**: when $\lambda_n \leq 1$, the error is $(1 - \frac{1}{2}\phi_G^2)^k$ where $\phi_G$ is the conductance of the graph $G$

# Proof sketch

$$Lp = (D - A)p = b \qquad \Rightarrow \qquad p = (D^{-1}A)p + D^{-1}b$$

Transition matrix: $P = D^{-1}A$

Jacobi's iteration: $p^{(k+1)} = Pp^{(k)} + D^{-1}b$

$\Rightarrow$ A fixed point $p$ is automatically a solution to $Lp = b$

Error at step $k$: $e^{(k)} := p - p^{(k)} = e_{\perp}^{(k)} + c^{(k)}\mathbf{1}$

What really matters is $e_{\perp}^{(k)}$; we do not care about $c^{(k)}$, since $L\mathbf{1} = \mathbf{0}$ !

# Proof sketch

If we unroll $e_\perp^{(k)}$ (calculation omitted) we get

$$e_\perp^{(k+1)} = \left( I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top \right) P^k e_\perp^{(k)}$$

i.e., $e_\perp^{(k+1)}$ equals $P^k e_\perp^{(k)}$ without its component parallel to $\mathbf{1}$

$\Rightarrow$ The behavior of the error is dictated by the eigenstructure of $P$

# Proof sketch

$P := D^{-1}A$ is similar to $\mathcal{N} := D^{-1/2}AD^{-1/2}$
(so $P^k$ is similar to $\mathcal{N}^k$)

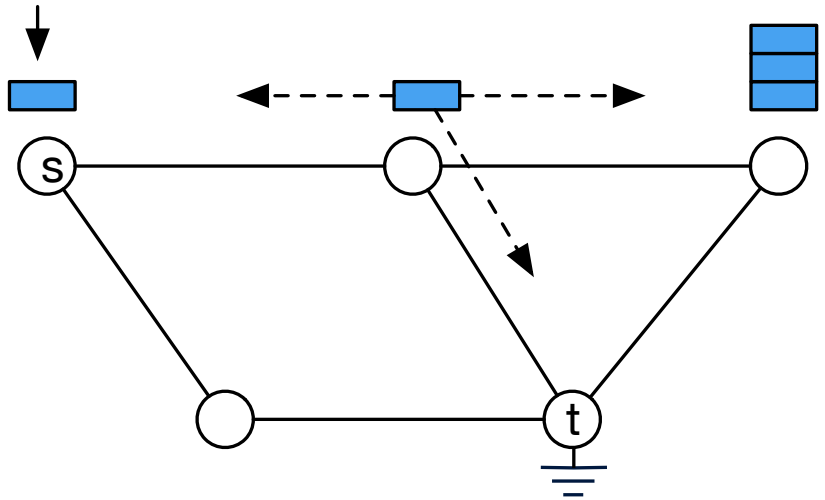
- Each eigenvector $u_i$ of $\mathcal{N}$ corresponds to an eigenvector $v_i$ of $P$ and vice versa, via $u_i = D^{1/2}v_i$
- $u_i$ and $v_i$ are associated to the same eigenvalue, call it $\rho_i$, of $\mathcal{N}$ and $P$
- $\rho_1 = 1 \geq \rho_2 \geq \ldots \geq \rho_n \geq -1$
- $\rho_i = 1 - \lambda_i$ since $\mathcal{L} = I - \mathcal{N}$

# Concluding the proof

$$
\begin{aligned}
\left\| e_\perp^{(k)} \right\| &= \left\| (I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top) P^k e_\perp^{(0)} \right\| \\
&= \left\| (I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top) D^{-1/2} N^k D^{+1/2} e_\perp^{(0)} \right\| \\
&= \left\| (I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top) D^{-1/2} (\sum_{i=2}^{n} \rho_i^k u_i u_i^\top) D^{+1/2} e_\perp^{(0)} \right\| \\
&\leq \left\| D^{-1/2} \right\| \left\| \sum_{i=2}^{n} \rho_i^k u_i u_i^\top \right\| \left\| D^{1/2} \right\| \left\| e_\perp^{(0)} \right\| \\
&\leq \sqrt{\frac{d_{\max}}{d_{\min}}} \max(|\rho_2|, |\rho_n|)^k \left\| e_\perp^{(0)} \right\| . \quad \square
\end{aligned}
$$

# A stochastic diffusion-based model

Instead of continuous flows, consider flow particles (tokens)




Repeat forever:

1. insert one new token at the source
2. each token moves from node $u$ to neighbor $v$ with probability proportional to the weight $x_{uv}$
3. remove all tokens at the sink, if any

# Why token diffusion?

Well-known analogy between the random walk and the electrical flow

**Theorem [e.g. Doyle-Snell (1984); Tetali (1991)]**

For a single random walker, and any node $u$,

$$\mathbb{E}[\#\text{ of visits to } u \text{ before reaching sink}] = d_u \cdot p_u$$

where $p$ is the solution to $Lp = b$ with $p_t = 0$

We use the same intuition but with many "staggered" random walks
$\Rightarrow$ Local instead of global estimator

# Correctness of token diffusion

## Theorem 2

For our token diffusion process, the number of tokens, $Z_u^{(k)}$, on node $u$ at time $k$ satisfies

$$\mathbb{E}\left[Z_u^{(k)}\right] \to d_u \cdot p_u \qquad \text{as } k \to \infty$$

where $p$ is the solution to $Lp = b$ with $p_t = 0$

$\Rightarrow$ Local number of tokens at $u$ can be used to estimate the local node potential $p_u$

We use $V_u^{(k)} := Z_u^{(k)}/d_u$ as an estimate of $p_u$

# Proof idea

We compare our estimator $V_u^{(k)} := Z_u^{(k)}/d_u$ against the $k$-th iterate of a modified Jacobi iteration:

$$p_u^{(0)} = 0 \text{ for all } u \in N,$$

$$p_u^{(k+1)} = \begin{cases} \frac{1}{d_u}\left(\sum_{v\sim u} x_{uv} p_v^{(k)} + b_u\right) & \text{if } u \neq t \text{ (sink)}, \\ 0 & \text{if } u = t \text{ (sink)}. \end{cases}$$

By the way diffusion is defined: $\mathbb{E}[V_u^{(k)}] = p_u^{(k)}$ (proof by induction)

Hence $\mathbb{E}[V_u^{(k)}] \to p_u$ as $k \to \infty$, if we can prove $p_u^{(k)} \to p_u$

# Proof idea

We can rewrite the iteration as

$$p^{(k+1)} = \underline{P} p^{(k)} + D^{-1} \underline{b}$$

with $\underline{P}$ and $\underline{b}$ obtained by zeroing out entries on row/column $t$ (sink).

## Lemma

*The spectral radius of $\underline{P}$, $\underline{\rho}$, satisfies*

$$\underline{\rho} = 1 - \sum_{i=1}^{n} v_i \cdot P_{i,t} < 1,$$

*where $v$ is the left dominant eigenvector of $\underline{P}$ (with $\|v\|_1 = 1$).*

Hence the iterations converge to a fixed point

# Time complexity

## Theorem 3 (Time complexity)

$$\left\|p^{(k)} - p\right\| \leq \left(1 - \frac{1}{8}\frac{\overline{d}_{\min}}{d_{\max}}\frac{d(t)}{\overline{d}(\overline{G})} \cdot \overline{\lambda}_2\right)^k$$

where

- $\overline{G}$ is $G - t$
- $\overline{d}(\cdot)$ is the volume in $\overline{G}$
- $\overline{\lambda}_2$ is the 2nd smallest eigenvalue of $\mathcal{L}(\overline{G})$

**Example.** If $G$ is regular,

$$\left\|p^{(k)} - p\right\| \leq \left(1 - \frac{\overline{\lambda}_2}{8n}\right)^k$$

# Message complexity

## Theorem 4 (Message complexity)

As $k \to \infty$, the expected message complexity per round of Token Diffusion is

$$O(n\, d_{\max}\, \mathcal{E})$$

where $\mathcal{E} = p^\top L p$ is the energy of the electrical flow.

# Stochastic accuracy of the estimator

Previous results concern expected values
Are the estimators accurate with high probability?

**Definition.** $X$ gives an $(\epsilon, \delta)$-approximation of $Y$ if

$$\Pr[|X - Y| > \epsilon Y] \leq \delta$$

## Lemma

*If we inject $M \geq 1$ tokens per round (instead of 1), then for any $k$, $u$ such that*

$$p_u^{(k)} \geq \frac{3}{\epsilon^2 M d_u} \ln \frac{2}{\delta},$$

*the estimator $V_u^{(k)}/M$ provides an $(\epsilon, \delta)$-approximation of $p_u^{(k)}$.*

# Summary

- Methods based on electrical flows can be used to design fast and conceptually simple network optimization algorithms
- Electrical flows were known to be effectively computable in a centralized setting
- We give decentralized methods and bound their time and message complexity as a function of network parameters

**Thanks for the attention!**

# Some references


- Spielman, Teng (2004)
  Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems, STOC
- Christiano et al. (2010)
  Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs, STOC
- B., Mehlhorn, Varma (2012)
  Physarum can compute shortest paths, SODA & *J.Theor. Biology* 309, pp. 121–133.
- Becchetti et al. (2013)
  Physarum can compute shortest paths: Convergence proofs and complexity bounds, ICALP
- Ma et al. (2013)
  Current-reinforced random walks for constructing transport networks, *Interface*
- Cohen et al. (2017)
  Negative-weight shortest paths and unit capacity minimum cost flow in $\tilde{O}(m^{10/7} \log W)$ time, SODA