Graphs Algorithms and Continuous Optimization

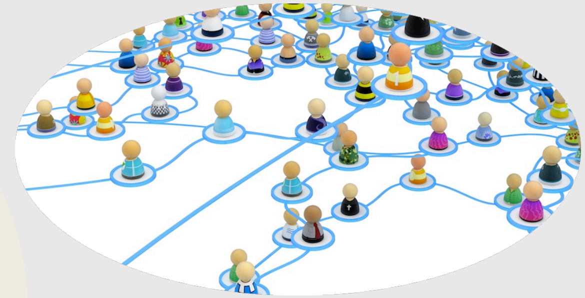
Part I: Overview

Aleksander Mądry

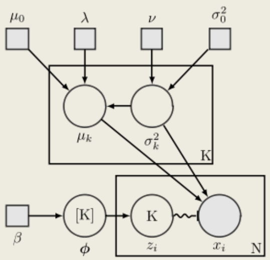
Graphs are everywhere

road/network routing

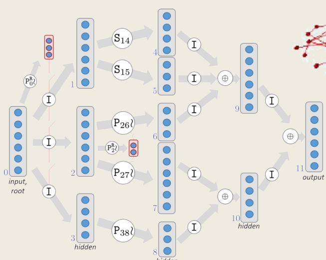
social sciences

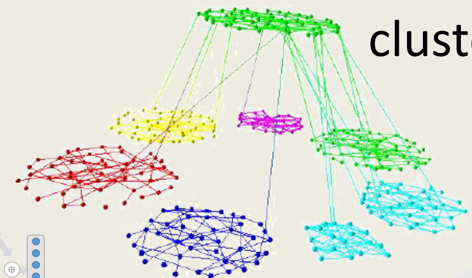


economics

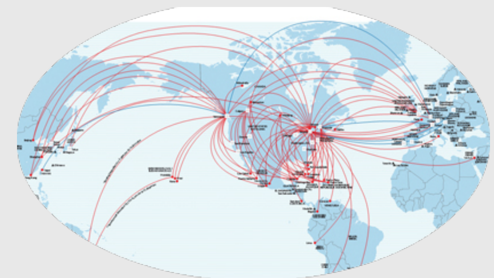


inference

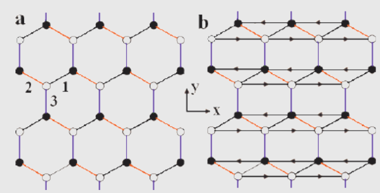




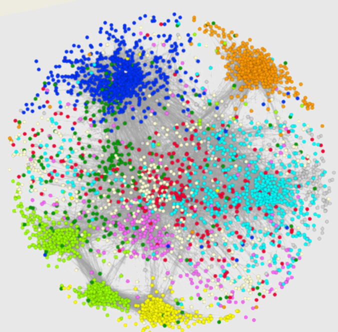
clustering



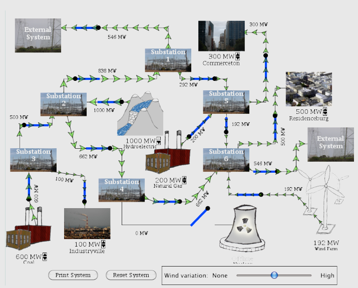
load balancing

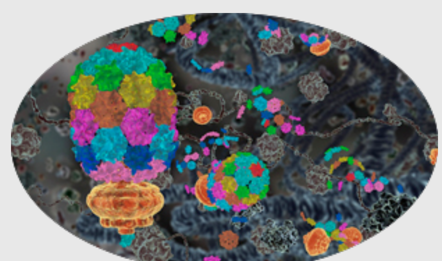


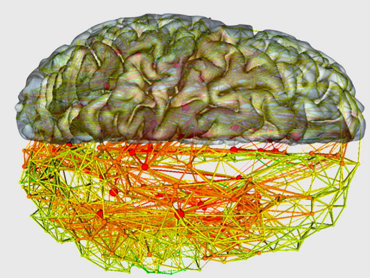
chemistry



biology





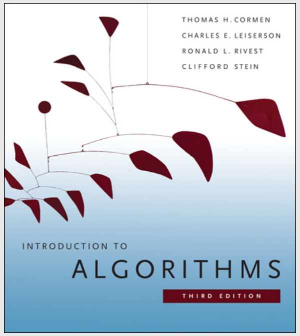


civil engineering

physics

neuroscience

Landscape of Algorithmic Graph Theory



→ Graph algorithms were shaping our understanding of **general** algs since 1950s

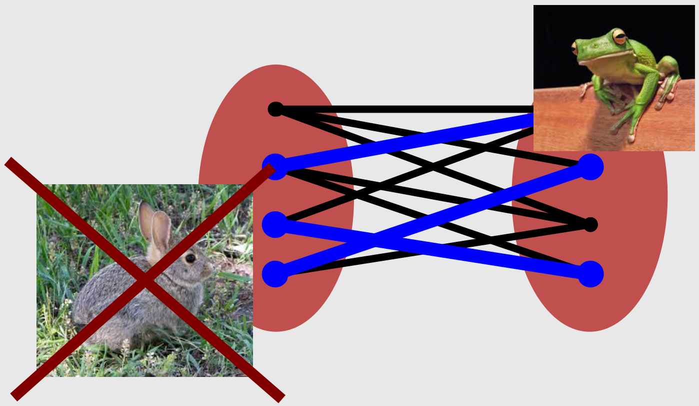
We have now a number of sophisticated algorithms for many graph problems

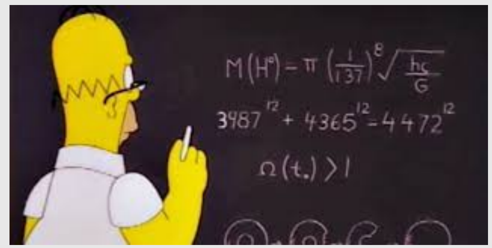
→ Most of these algs are **hand-crafted & combinatorial**
(“combinatorial” = manipulating paths, trees, cuts, partitions,…)

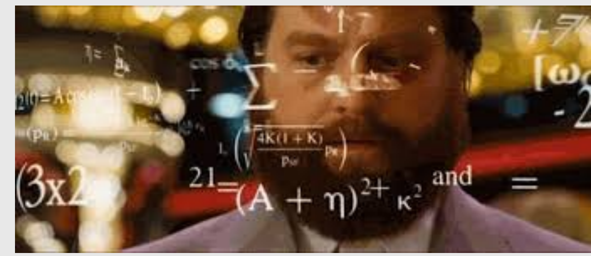
→ This is only natural!

Dominating mindset: “combinatorial” = “fast”

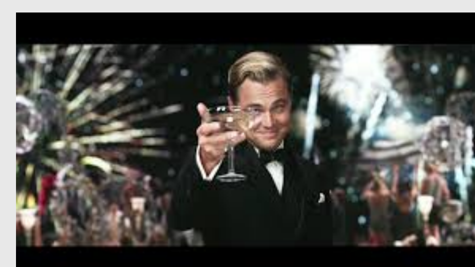
“Typical” graph algorithm production cycle (dramatized)







→ Define your problem
(e.g., min-cost rabbit-free matching in frog-like graphs)
→ Think hard…
→ …think some more…
→ ...think even more...
→ ...until you come up with a slick algorithm (and analysis)!



But…is it really the right approach?

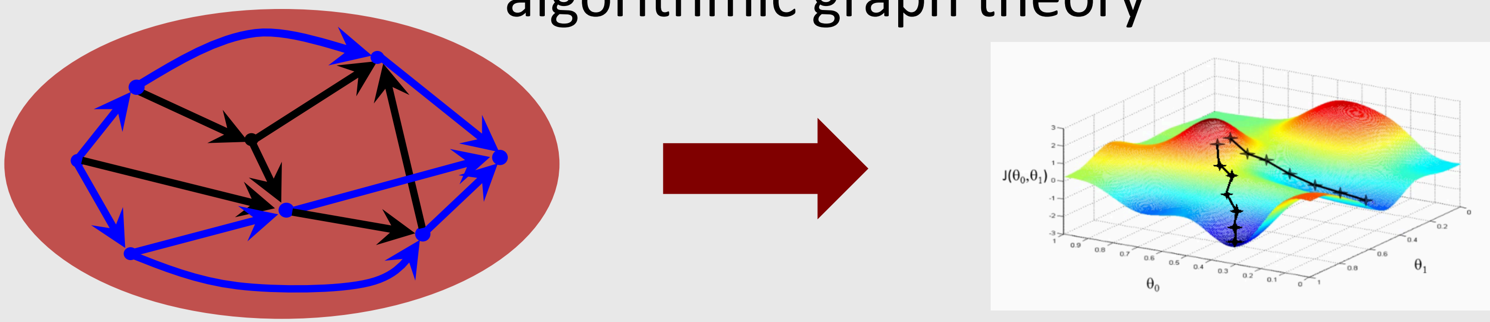
It works great if you are a graph algorithms aficionado
but not so much if you just want to solve the problem

→ **Often:** “combinatorial” = “ad hoc/hard to analyze”
→ You need to be/work with an **expert**
→ It is **hard** to come up with a new algorithm (trust me…)
→ **No robustness:** slight change/generalization of the problem might require a completely different algorithm
→ You need to do **implementation/deployment** from scratch

Does it have to be this way?

Overarching goal: Developing a unified and principled approach to graph algorithms

Specifically: Make continuous optimization the language of algorithmic graph theory



→ Formulate graph problem as continuous optimization task
(constrained minimization problem, linear program, SDP,…)

→ Apply an "off-the-shelf" algorithm to that formulation
(gradient descent, interior-point method, linear system solver,…)

→ If needed: add problem-specific insights to get optimal performance

Conventional wisdom: This will not work!
(Graphs are "inherently" combinatorial and general cont. opt. tools are too slow)

But: This already **did** work (multiple times!)
→ Most current fastest matching/flow algorithms follow this paradigm

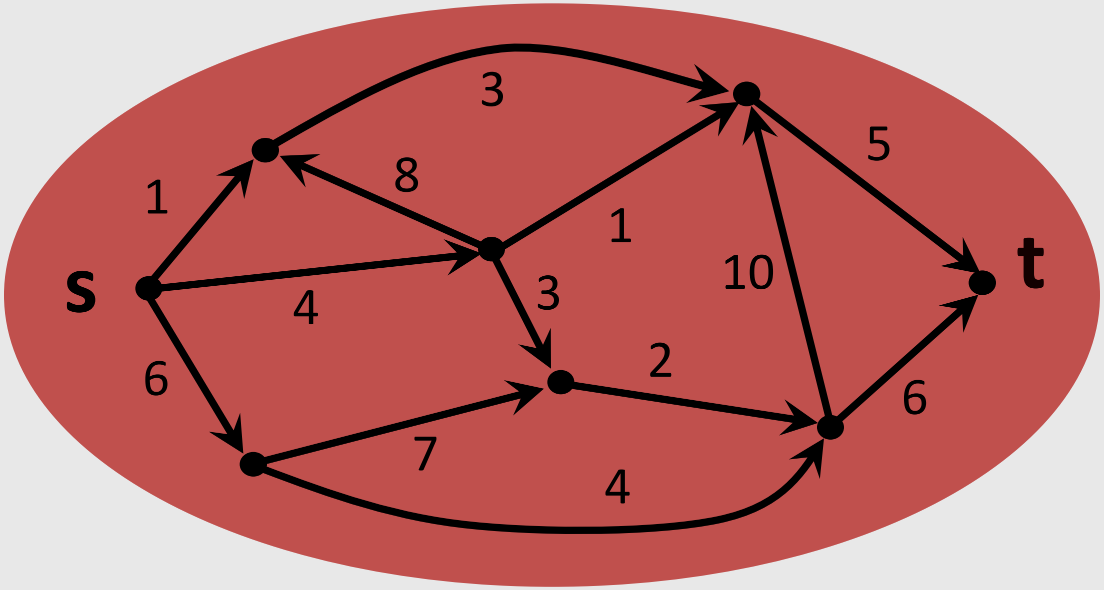
Our plan for this week:
Illustrate this theme on an example of a single problem

Problem: (Unit capacity) Maximum flow

Underlying approach:
Relate combinatorial structure of a graph to analytic/linear-algebraic properties of associated matrices

Maximum flow problem

Input: Directed graph **G**,
integer **capacities u_e**,
source s and **sink t**



Think: arcs = roads
capacities = # of lanes
s/t = origin/destination

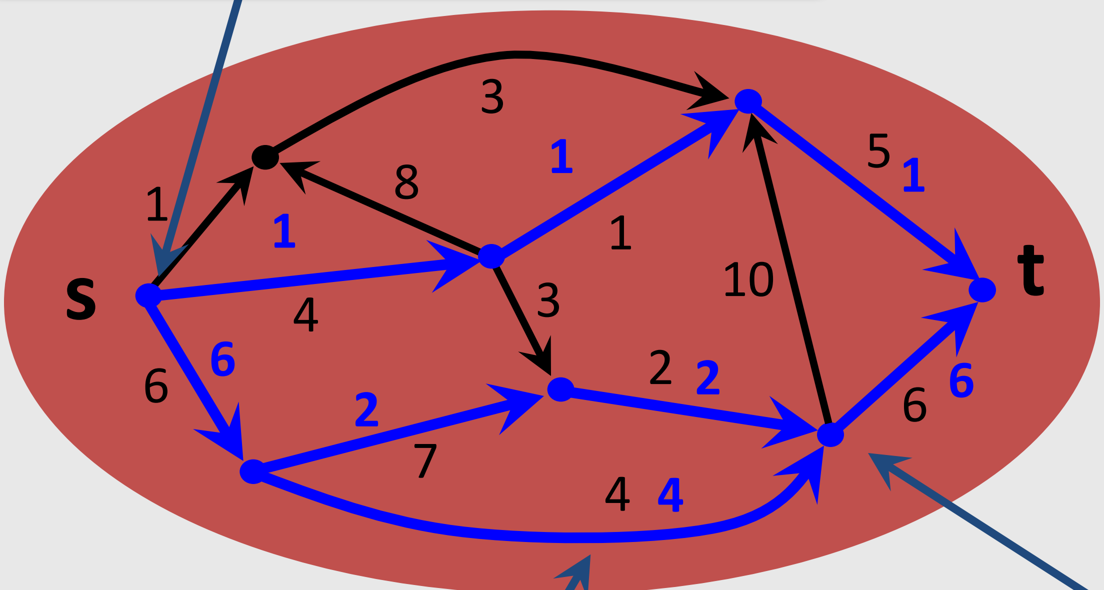
Task: Find a **feasible s-t flow** of **max value**

(**Think:** Estimate the **max** possible rate of traffic from **s** to **t**)

Maximum flow problem

Input: Directed graph **G**, integer **capacities u_e**, **source s** and **sink t**

value = net flow out of **s**



Think: arcs = roads
capacities = # of lanes
s/t = origin/destination

Max flow value
F*=10

no overflow on arcs:
0 ≤ f(e) ≤ u(e)

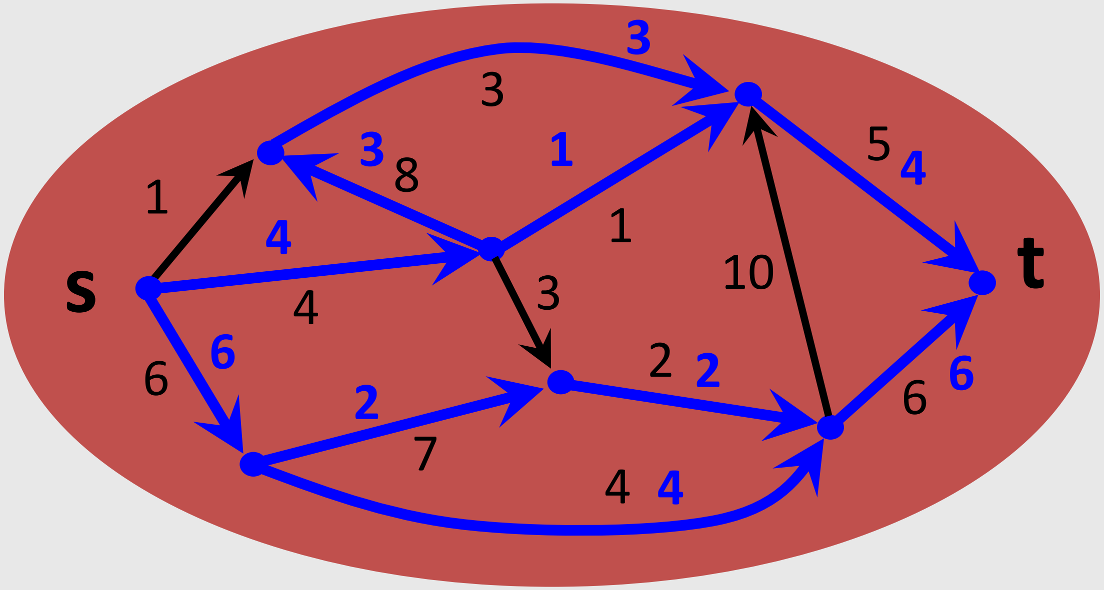
no leaks at all **v≠s,t**

Task: Find a **feasible s-t flow** of **max value**

(**Think:** Estimate the **max** possible rate of traffic from **s** to **t**)

Maximum flow problem

Input: Directed graph **G**, integer **capacities u_e**, **source s** and **sink t**



Think: arcs = roads
capacities = # of lanes
s/t = origin/destination

Max flow value
F*=10

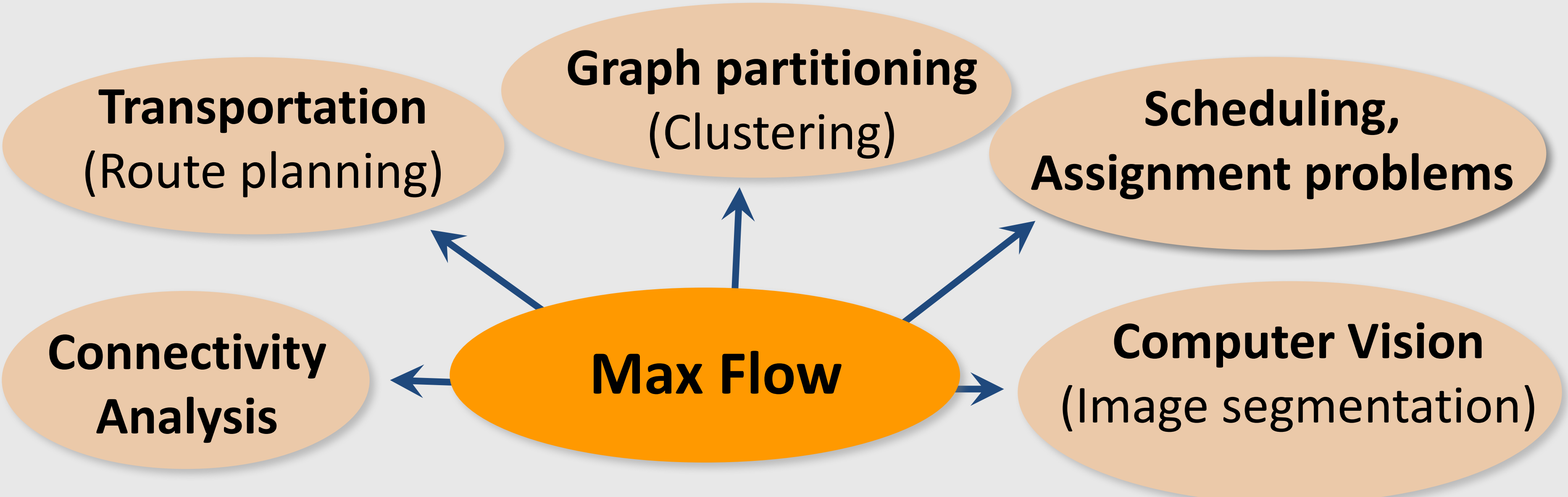
Task: Find a **feasible s-t flow** of **max value**

(**Think:** Estimate the **max** possible rate of traffic from **s** to **t**)

Why is this a good problem to study?

Max flow is a fundamental optimization problem

- **Extensively studied since 1930s** (classic 'textbook problem')
- **Surprisingly diverse set of applications**
- **Very influential in development of (graph) algorithms**



A (Very) Brief History of Max Flow

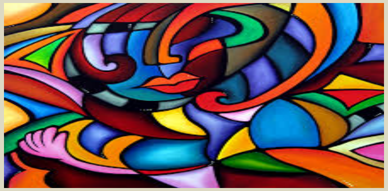
Our focus today: Sparse graphs, i.e., **m=O(n)**
(This setting is a great benchmark for combinatorial graph alg.)

"Classical" Era:

→ Uses purely **combinatorial** algorithms
→ [ET '75, K '73]: $O(n^{3/2})$ time for unit-capacities (**U=1**)
→ [GR '98]: $\tilde{O}(n^{3/2} \log U)$ time for general case

Emerging barrier: $\Omega(n^{3/2})$ run time

"Modern" Era:



→ Relies on linear algebra and **continuous** opt.
→ [M '10, CKMST '11, LRS '13, S '13, KLOS '14, P '14]: $\tilde{O}(n\varepsilon^{-2})$ time for **undirected** and **(1+ε)-approx.**
→ [M '13]: $\tilde{O}((nU)^{10/7})$ time – improvement for **U=1**

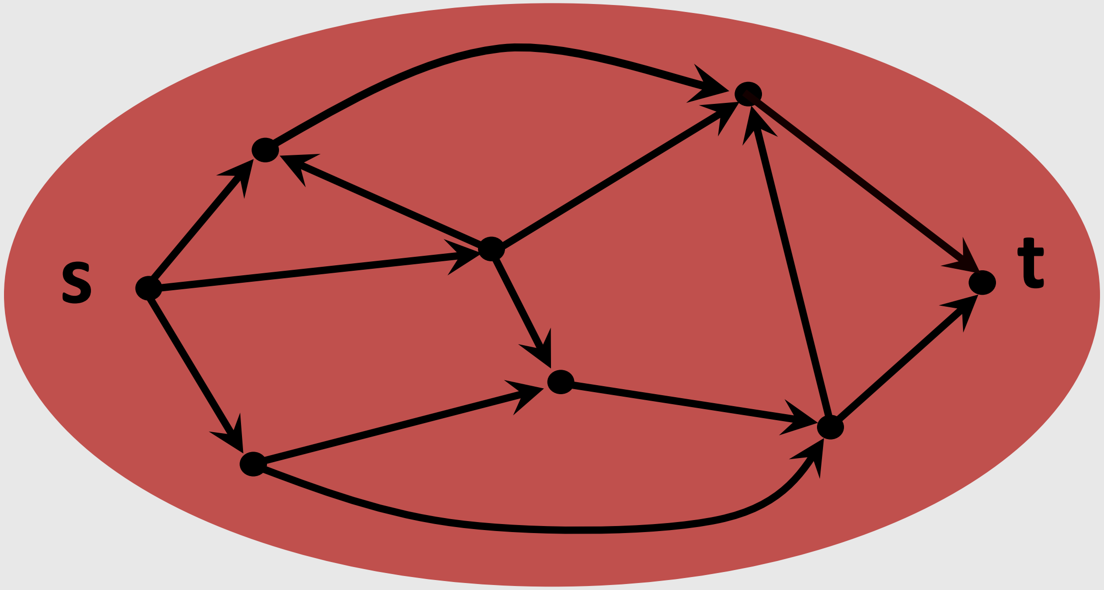
These lectures: A glimpse of how continuous opt. comes into play here

Classic approach

Augmenting paths framework (for U=1)

[Ford Fulkerson '56]

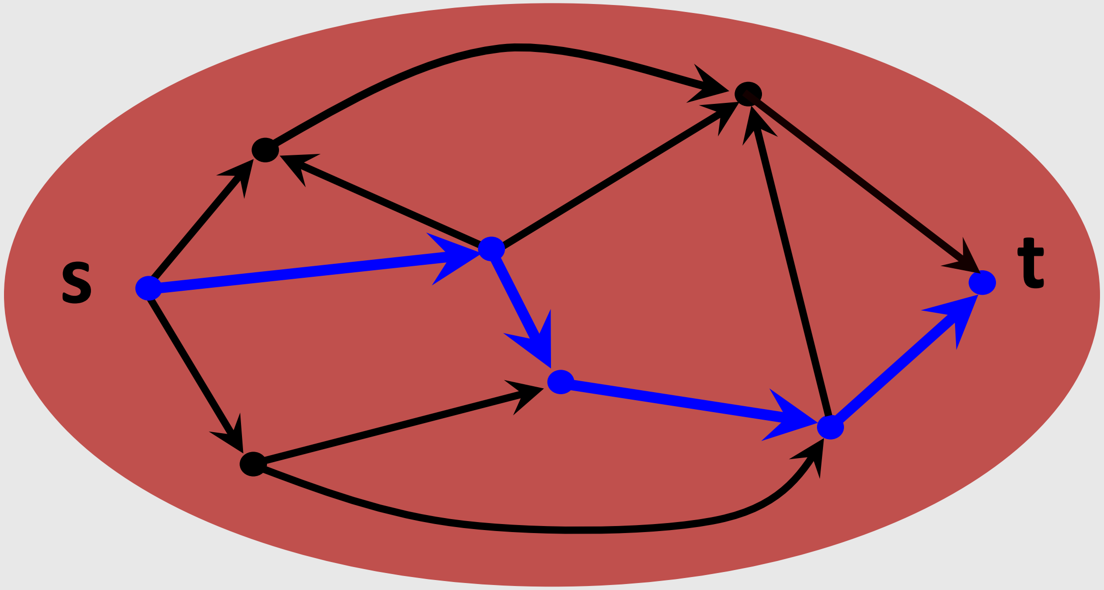
Basic idea: Repeatedly find **s-t paths** in the **residual graph**



Augmenting paths framework (for U=1)

[Ford Fulkerson '56]

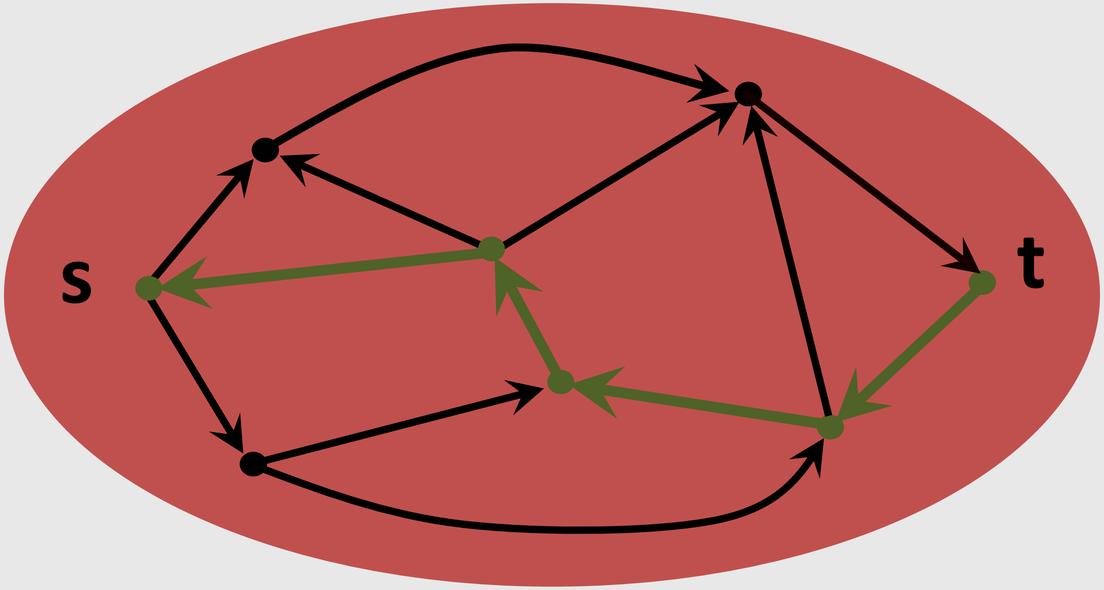
Basic idea: Repeatedly find **s-t paths** in the **residual graph**



Augmenting paths framework (for U=1)

[Ford Fulkerson '56]

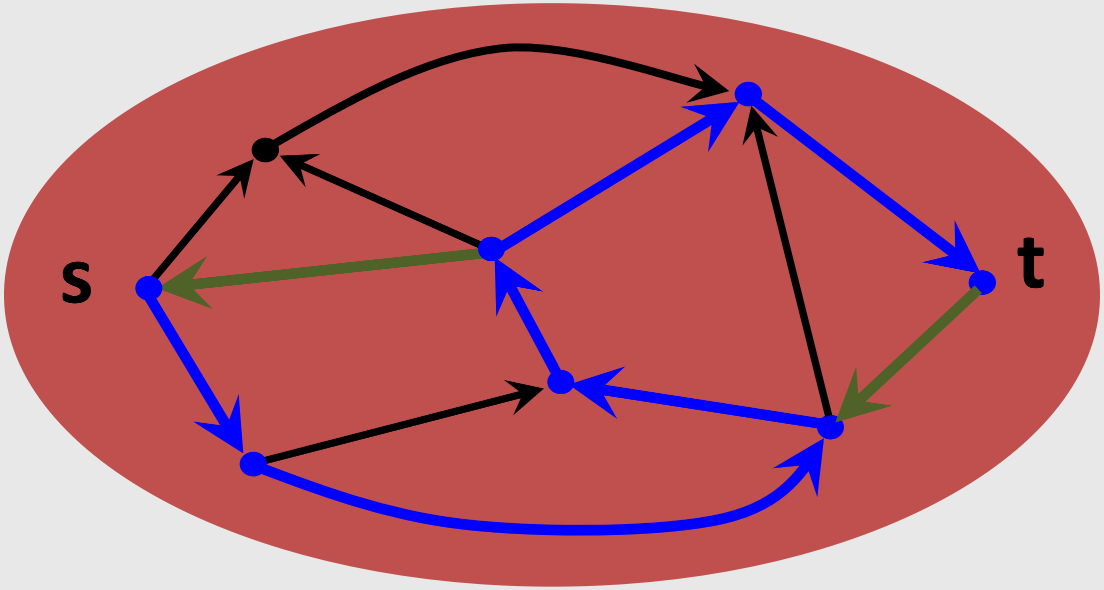
Basic idea: Repeatedly find **s-t paths** in the **residual graph**



Augmenting paths framework (for U=1)

[Ford Fulkerson '56]

Basic idea: Repeatedly find **s-t paths** in the **residual graph**

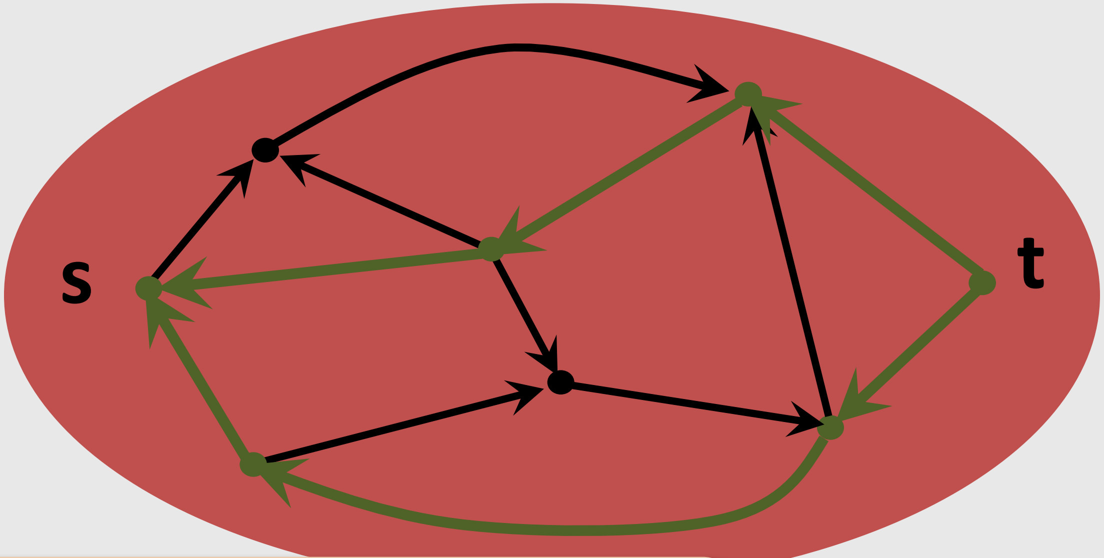


Augmenting paths framework (for U=1)

[Ford Fulkerson '56]

Basic idea: Repeatedly find **s-t paths** in the **residual graph**

Advantage: Simple, purely combinatorial and greedy (flow is built path-by-path)



Problem:
Very difficult to analyze

Naïve impl

(≤ **n** augme   path)

Unclear how to get
a further speed-up via this route

Sophisticated implementation and arguments:

O($n^{3/2}$) time [Karzanov '73] [Even Tarjan '75]

Beyond augmenting paths

For now: Focus on the **undirected** variant

How to approach max flow via continuous opt.?

First: View a flow **f** as a vector

→ **f** is an **m**-dim vector with $|f_e|$ being the amount of flow on **e** and **sign**(f_e) encoding its direction (wrt fixed edge orientation)

Then: Phrase (undir.) max flow as an continuous opt. problem

Max flow formulation:

Find **f** that minimizes $\max_e |f_e|$
among all **s-t** flows **f** of value **1**

How to approach max flow via continuous opt.?

First: View a flow **f** as a vector

→ **f** is an **m**-dim vector with $|f_e|$ being the amount of flow on **e** and $\mathbf{sign(f_e)}$ encoding its direction (wrt fixed edge orientation)

Then: Phrase (undir.) max flow as an continuous opt. problem

Max flow formulation:

Find **f** that minimizes $\boxed{\max_e |f_e|}$ $= |f|_\infty$
among all **s-t** flows **f** of value **1**

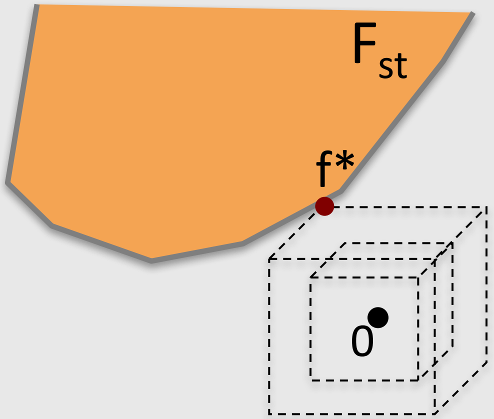
Note: The optimum value of the objective (**ℓ_∞-norm**) here is **1/F***

Finally: Use **standard** cont. opt. tools to solve this problem!

Max flow: Minimize $|\mathbf{f}|_\infty$
s.t. **f** being an **s-t** flow of value **1**

How to solve this problem?

$\mathbf{F_{st}}$ = space of all unit s-t flows



Max flow: Minimize $|\mathbf{f}|_\infty$
s.t. **f** being an **s-t** flow of value **1**

Canonical alg.: (Sub)gradient descent method
(= “continuous greedy” strategy)

step size $\eta>0$

(Sub)gradient descent (SGD):

→ Start with some arbitrary unit s-t flow **f**

→ In each step:

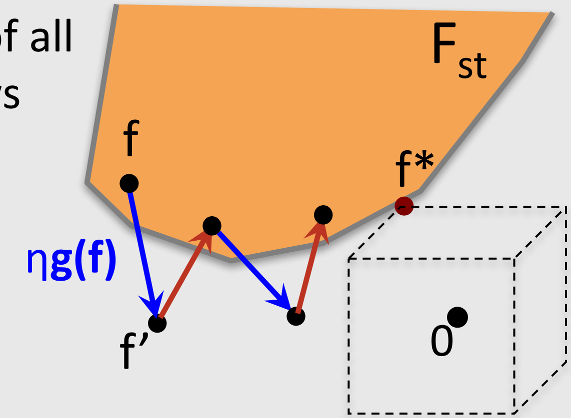
- Set **f’ = f –** η **g(f)**
  (**Note: f’** might **not** be a unit s-t flow)
- Set **f = Π(f’)** [projection back on $\mathbf{F}_{st}$]

→ After **T** steps, return **f**

That’s it! **Only nontrivial element:**
Computing the projection **Π(f’)**

(Incidentally) we know how to do that already!

$\mathbf{F}_{st}$ = space of all unit s-t flows



g(f) = (sub)gradient at point **f** (“steepest increase direction”)

Here: g(f)≈argmax $\mathbf{f}_e$

Π(f’) = Projection on the space of unit **s-t** flows $\mathbf{F}_{st}$

Π(f’) = unit **s-t** flow **h’** that min. $|\mathbf{f'}-\mathbf{h'}|_2$

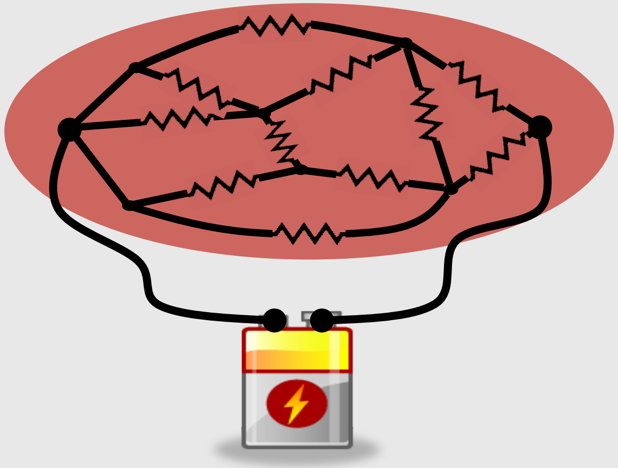
ℓ_2- projection on the space of s-t flows

Want to compute: $\Pi(f')$ = unit **s-t** flow **h'** that minimizes **$|f'-h'|_2$**

Answer: Electrical flows

Input: Undirected graph **G**, **resistances r_e**, **s** and **t**

Electrical flow of value F:
The unique minimizer of the **energy**
$$\mathbb{E}(f) = \Sigma_e\, r_e\, f(e)^2$$
among all **s-t** flows **f** of value **F**



Electrical flow computation

$$L\ x = b$$

Solving linear systems in graph Laplacian

Can compute elect. flow in nearly-linear time
[ST ’04, KMP ’10, KMP ’11, KOSZ ’13, LS ’13, CKPPR ’14, KLPSS ’16, KS ‘16]

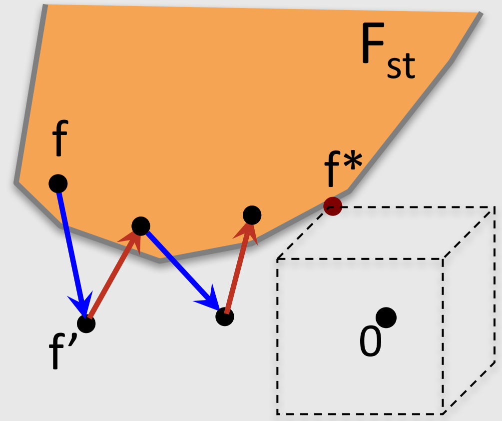
(Sub)gradient descent (SGD):

→ Start with some arbitrary unit s-t flow **f**

→ In each step:

- Set **f' = f –** η **g(f)**
  (**Note: f'** might **not** be a unit s-t flow)
- Set **f =** Π**(f')** [projection back on $\mathbf{F_{st}}$]

→ After **T** steps, return **f**



Standard SGD analysis:

After $\tilde{O}(n^2\varepsilon^{-2})$ steps, **f** is an **(1+ε)**-approx. max flow

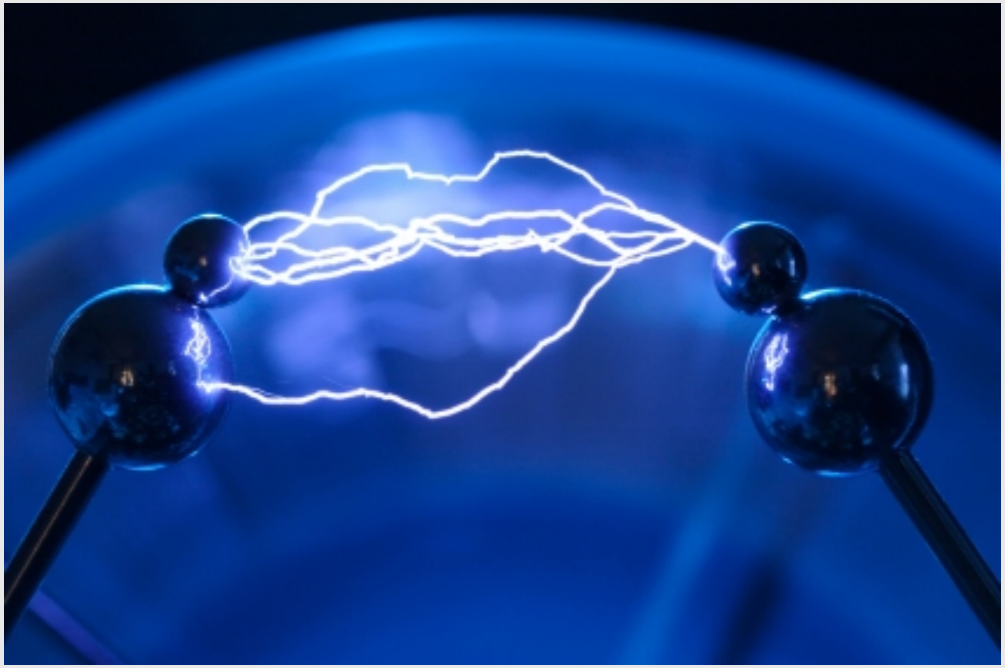
Resulting running time: $\tilde{O}(n^3\varepsilon^{-2})$

Not great, but:

→ Our approach is principled/robust

→ This is just a generic/"off-the-shelf" attempt

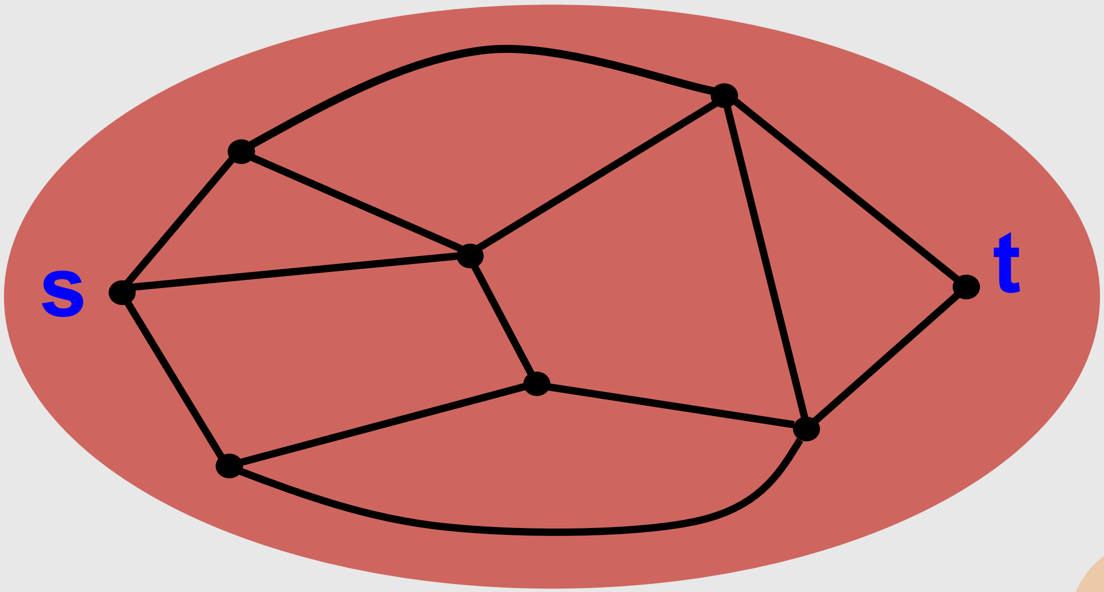
As we will see: Cont. opt. has much more to offer here!



Electrical Flows

Electrical flows (Take I)

Input: **Undirected** graph **G**,
resistances r_e,
source s and **sink t**

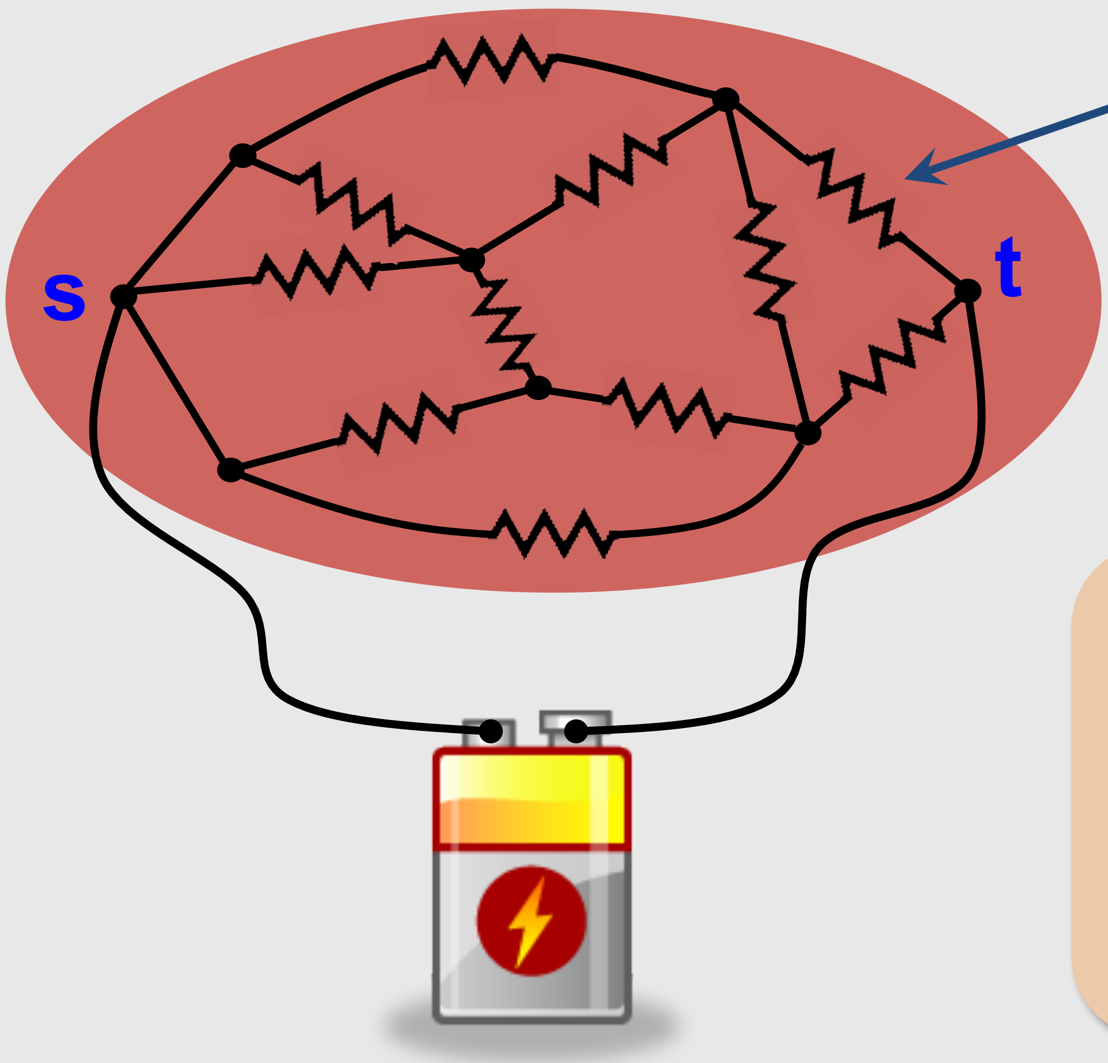


Recipe for elec. flow:

1) Treat edges as **resistors**

Electrical flows (Take I)

Input: **Undirected** graph **G**,
resistances r_e,
source s and **sink t**



resistance r_e

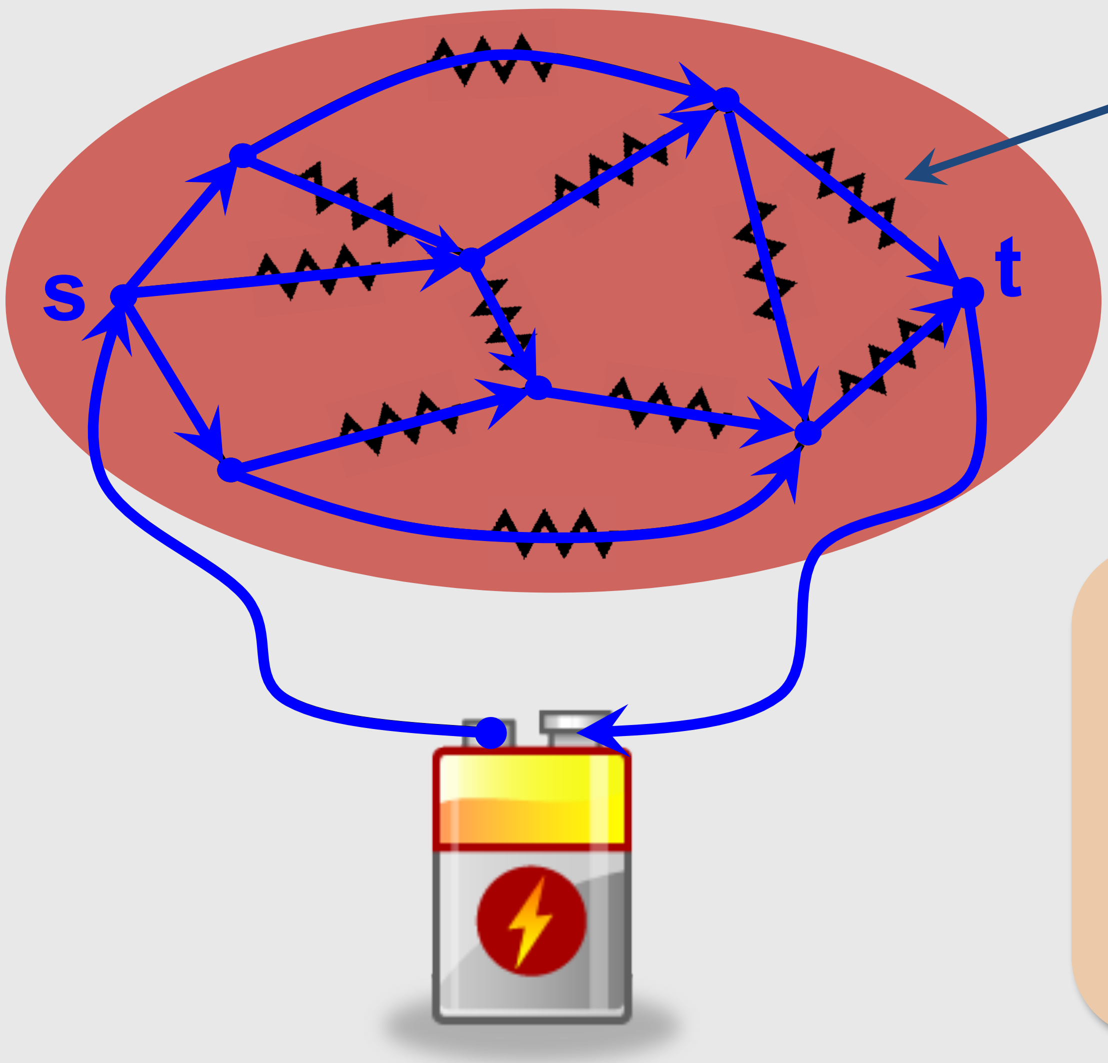
Recipe for elec. flow:

1) Treat edges as **resistors**

2) Connect a **battery** to **s** and **t**

Electrical flows (Take I)

Input: **Undirected** graph **G**,
resistances r_e,
source s and **sink t**



resistance r_e

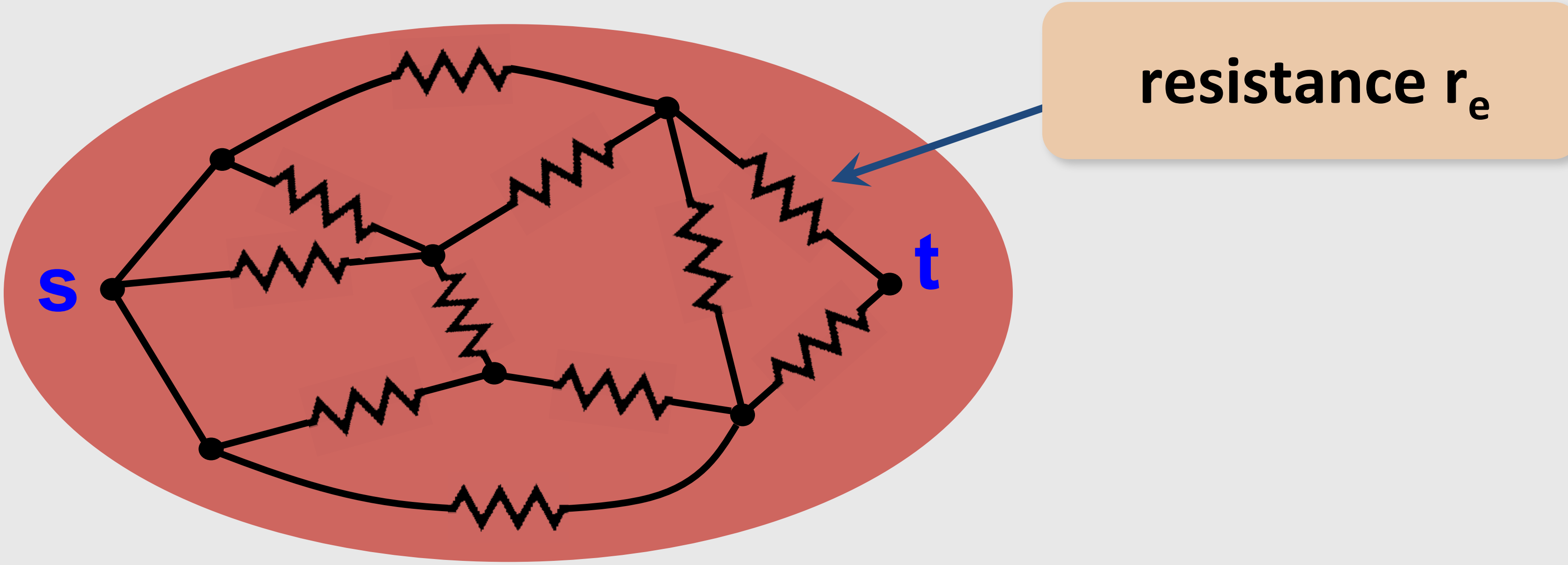
Recipe for elec. flow:

1) Treat edges as **resistors**

2) Connect a **battery** to **s** and **t**

Electrical flows (Take II)

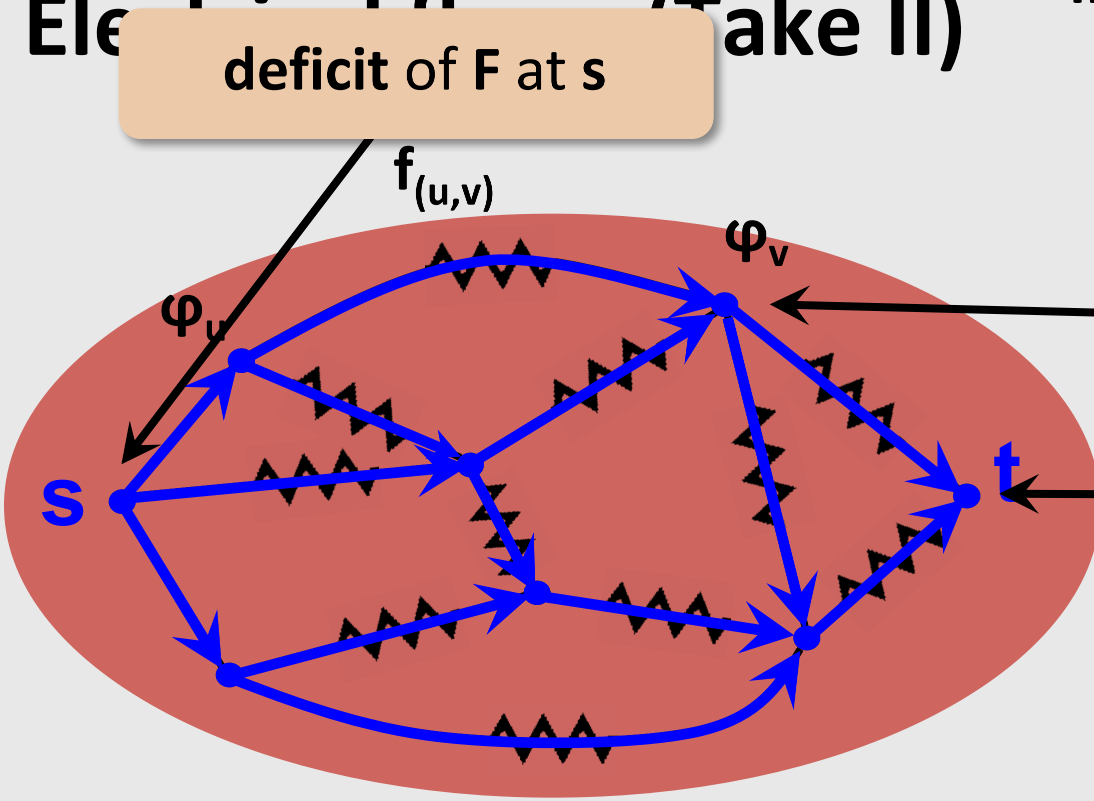
Input: **Undirected** graph **G**, **resistances r_e**, **source s** and **sink t**



(Another) recipe for electrical flow (of value F):

Ele… (Take II)

Input: **Undirected** graph **G**,
resistances r_e,
source s and **sink t**



(Another) recipe for electrical flow (of value F):

Find **vertex potentials** φ_v such that setting, for all **(u,v)**

$$f_{(u,v)} \leftarrow (\varphi_v - \varphi_u)/r_{(u,v)} \quad \textbf{(Ohm's law)}$$

gives a **valid s-t flow** of **value F**

Electrical flows (Take III)

Input: **Undirected** graph **G**, **resistances** $\mathbf{r_e}$, **source s** and **sink t**

Principle of least energy

Electrical flow of value F:
The unique minimizer of the **energy**

$$\mathbb{E}(f) = \Sigma_e \, r_e \, f(e)^2$$

among all **s-t** flows **f** of value **F**

Electrical flows = ℓ_2-minimization

How to compute an electrical flow?

Input: Graph **G=(V,E)**, **resistances r_e**, **source s** and **sink t, value F=1**

Solve a linear system!

Wlog as elect. flow are invariant under scaling

How to compute an electrical flow?

Input: Graph **$G=(V,E)$**, **resistances r_e**, **source s** and **sink t**, **value $F=1$**

Solve a linear system!

Observe: It suffices to compute **vertex potentials φ_v**

Ohm's law: If φ is an (**$|V|$-dim**) vector of **vertex potentials** then

$$f = R^{-1}B^T \varphi$$

is the corresponding flow

Here:

→ **f** is an **$|E|$**-dim vector with **$|f_e|$** giving the amount of flow on **e** and **sign(f_e)** encoding its direction (wrt edge orientation)

→ **R** is an **$|E| \times |E|$ diagonal** matrix with **$R_{ee} = r_e$**

→ **B** is an **$|V| \times |E|$** matrix with **e-th** column, for **$e=(v,u)$**, having **-1** (resp. **+1**) at its **v-th** (resp. **u-th**) coordinate and **0** everywhere else

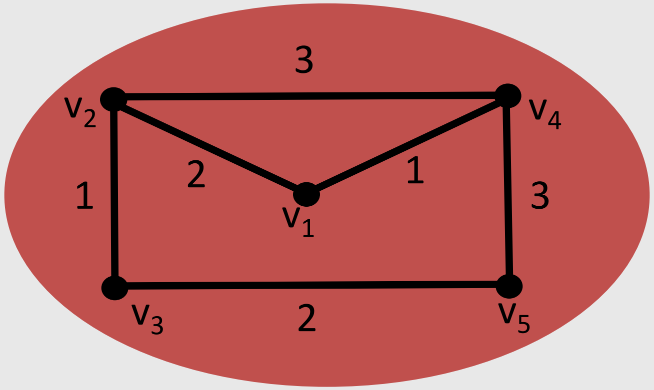
How to compute an electrical flow?

Ohm's law: If **φ** is an (**|V|-dim**) vector of **vertex potentials** then

$$\mathbf{f} = \mathbf{R}^{-1}\mathbf{B}^{T}\,\boldsymbol{\varphi}$$

is the corresponding flow

Example:



$$\mathbf{B} = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

|V|=5, |E|=6, all edges oriented (v_i, v_j) with $i<j$

$$\mathbf{R} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

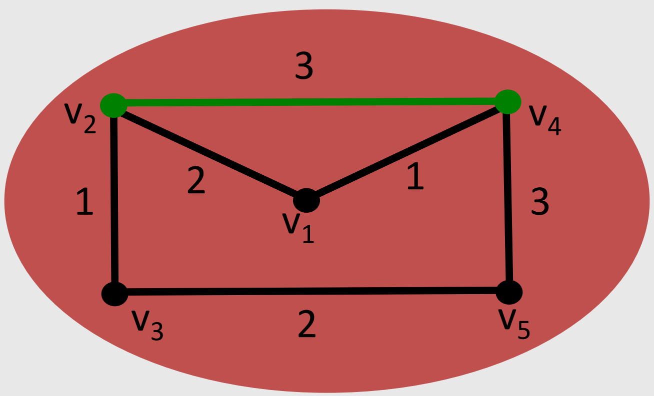
How to compute an electrical flow?

Ohm's law: If **φ** is an (**|V|-dim**) vector of **vertex potentials** then

$$\mathbf{f = R^{-1}B^{T}\,\varphi}$$

is the corresponding flow

Example:



|V|=5, |E|=6, all edges oriented **(v_i,v_j)** with **i<j**

$$B = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \boxed{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

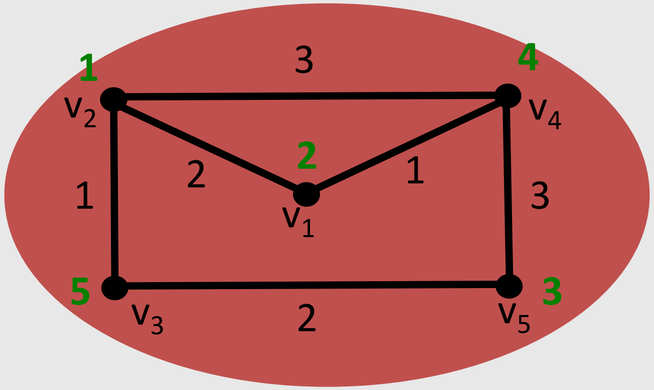
How to compute an electrical flow?

Ohm's law: If **φ** is an (**|V|-dim**) vector of **vertex potentials** then

$$\mathbf{f = R^{-1}B^{T}\,\varphi}$$

is the corresponding flow

Example:



$$\mathbf{B} = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

|V|=5, |E|=6, all edges oriented $(\mathbf{v_i, v_j})$ with **i<j**

$$\boldsymbol{\varphi} = \begin{bmatrix} 2 \\ 1 \\ 5 \\ 4 \\ 3 \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

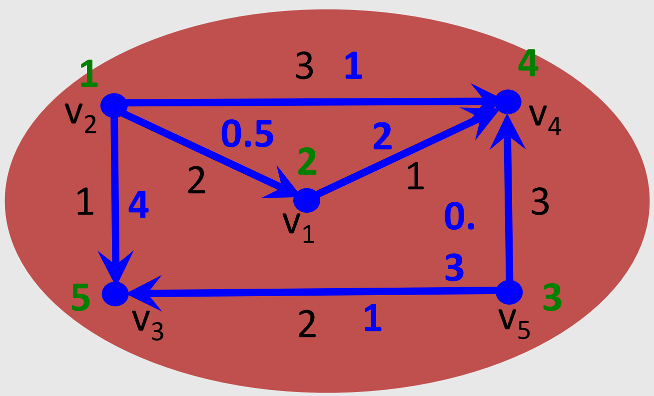
How to compute an electrical flow?

Ohm's law: If φ is an (**|V|-dim**) vector of **vertex potentials** then

$$\mathbf{f = R^{-1}B^T \varphi}$$

is the corresponding flow

Example:



$$B = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

|V|=5, |E|=6, all edges oriented $(\mathbf{v_i,v_j})$ with **i<j**

$$R = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

$$\varphi = \begin{bmatrix} 2 \\ 1 \\ 5 \\ 4 \\ 3 \end{bmatrix} \xrightarrow{R^{-1}B^T} f = \begin{bmatrix} -0.5 \\ 2 \\ 4 \\ 1 \\ -1 \\ -0.3 \end{bmatrix}$$

How to compute an electrical flow?

Ohm's law: If $\boldsymbol{\varphi}$ is an (**|V|-dim**) vector of **vertex potentials** then

$$\mathbf{f} = \mathbf{R}^{-1}\mathbf{B}^{T}\,\boldsymbol{\varphi}$$

is the corresponding flow

Recall: $\boldsymbol{\varphi}$ induces an electrical flow **f** iff

f is a valid **s-t** flow
(i.e., satisfies flow conservation constraints)

Equivalently: $\boldsymbol{\varphi}$ induces an electrical flow **f** iff

$$\mathbf{B}\,\mathbf{f} = \boldsymbol{\chi}_{s,t}$$

where $\boldsymbol{\chi}_{s,t}$ has a **1** at **t**, **-1** at **s** and **0**s everywhere else

Note: $(\mathbf{Bf})_v$ is the excess/deficit of **f** at **v**

How to compute an electrical flow?

Ohm's law: If **φ** is an (**|V|-dim**) vector of **vertex potentials** then

$$\mathbf{f} = R^{-1}B^{T}\varphi$$

is the corresponding flow

Recall: **φ** induces an electrical flow **f** iff

f is a valid **s-t** flow
(i.e., satisfies flow conservation constraints)

Equivalently: **φ** induces an electrical flow **f** iff

$$B\,\mathbf{f} = \chi_{s,t}$$

where $\chi_{s,t}$ has a **1** at **t**, **-1** at **s** and **0**s everywhere else

Note: $(\mathbf{Bf})_v$ is the excess/deficit of **f** at **v**

Putting it together: **φ** induces an electrical flow iff

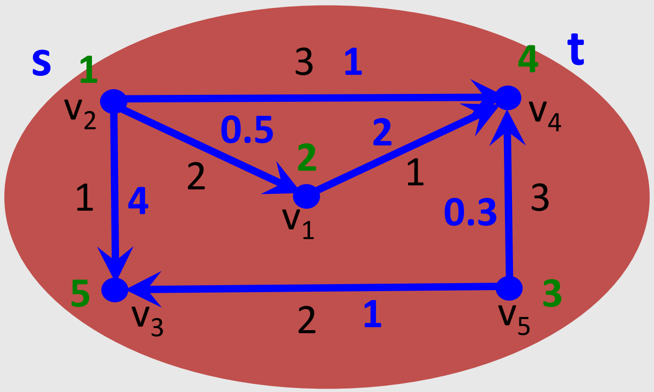
$$B\,R^{-1}B^{T}\varphi = \chi_{s,t}$$

How to compute an electrical flow?

Putting it together: φ induces an electrical flow iff

$$B\,R^{-1}B^T\,\varphi = \chi_{s,t}$$

Example:



$$B = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$|V|=5$, $|E|=6$, all edges oriented (v_i,v_j) with $i<j$

$$R = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

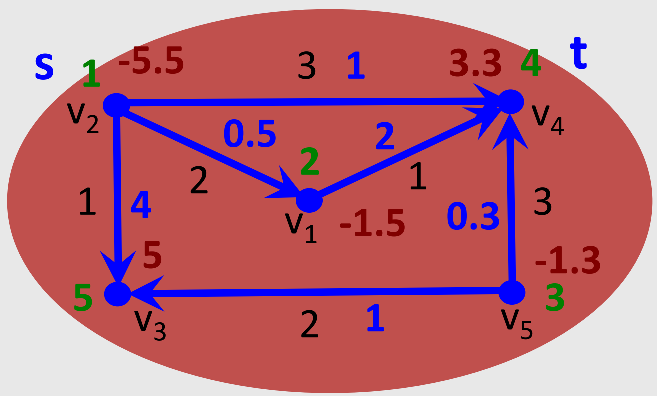
$$\varphi = \begin{bmatrix} 2 \\ 1 \\ 5 \\ 4 \\ 3 \end{bmatrix} \xrightarrow{R^{-1}B^T} f = \begin{bmatrix} -0.5 \\ 2 \\ 4 \\ 1 \\ -1 \\ -0.3 \end{bmatrix}$$

How to compute an electrical flow?

Putting it together: φ induces an electrical flow iff

$$\mathbf{B\,R^{-1}B^{T}\,\varphi = \chi_{s,t}}$$

Example:



$$B = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

|V|=5, |E|=6, all edges oriented **(v_i,v_j)** with **i<j**

$$\varphi = \begin{bmatrix} 2 \\ 1 \\ 5 \\ 4 \\ 3 \end{bmatrix} \xrightarrow{R^{-1}B^{T}} f = \begin{bmatrix} -0.5 \\ 2 \\ 4 \\ 1 \\ -1 \\ -0.3 \end{bmatrix} \xrightarrow{B} \begin{bmatrix} -1.5 \\ -5.5 \\ 5 \\ 3.3 \\ -1.3 \end{bmatrix} \neq \underbrace{\begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \\ 0 \end{bmatrix}}_{=\,\chi_{s,t}}$$

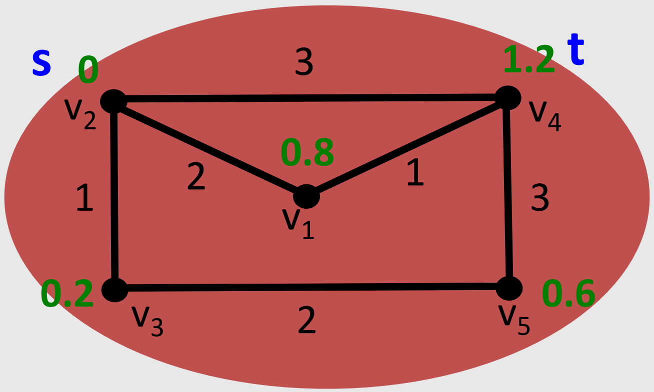
$$R = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

How to compute an electrical flow?

Putting it together: φ induces an electrical flow iff

$$B\,R^{-1}B^{T}\,\varphi = \chi_{s,t}$$

Example:



|V|=5, |E|=6, all edges oriented **(v_i,v_j)** with **$i<j$**

$$B = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\varphi = \begin{bmatrix} 0.8 \\ 0 \\ 0.2 \\ 1.2 \\ 0.6 \end{bmatrix}$$

$$\chi_{s,t} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

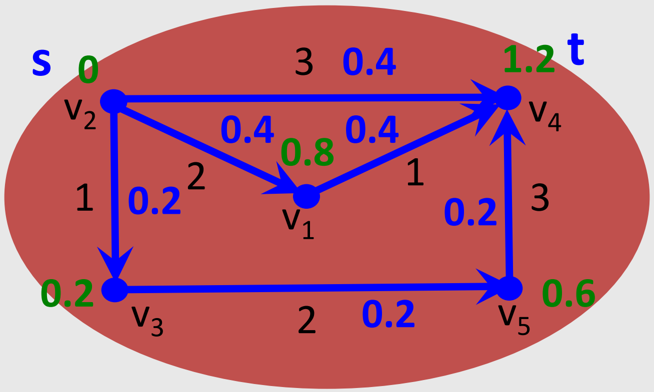
$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

How to compute an electrical flow?

Putting it together: φ induces an electrical flow iff

$$B\, R^{-1} B^T \varphi = \chi_{s,t}$$

Example:



$$B = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$|V|=5$, $|E|=6$, all edges oriented (v_i, v_j) with $i<j$

$$\varphi = \begin{bmatrix} 0.8 \\ 0 \\ 0.2 \\ 1.2 \\ 0.6 \end{bmatrix} \xrightarrow{R^{-1}B^T} f = \begin{bmatrix} -0.4 \\ 0.4 \\ 0.4 \\ 0.2 \\ 0.2 \\ -0.2 \end{bmatrix}$$

$$\chi_{s,t} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

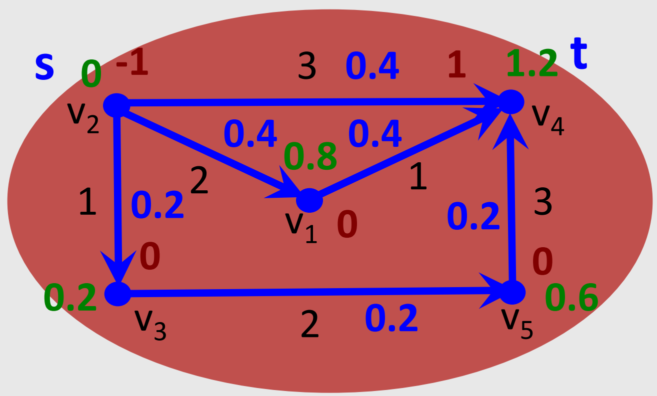
$$R = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

How to compute an electrical flow?

Putting it together: φ induces an electrical flow iff

$$B\, R^{-1} B^{T} \varphi = \chi_{s,t}$$

Example:



$$B = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

|V|=5, |E|=6, all edges oriented (v_i,v_j) with **i<j**

$$\varphi = \begin{bmatrix} 0.8 \\ 0 \\ 0.2 \\ 1.2 \\ 0.6 \end{bmatrix} \xrightarrow{R^{-1}B^{T}} f = \begin{bmatrix} -0.4 \\ 0.4 \\ 0.4 \\ 0.2 \\ 0.2 \\ -0.2 \end{bmatrix} \xrightarrow{B} \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \chi_{s,t} \checkmark$$

$$R = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

How to compute an electrical flow?

Bottom line:

Electrical flow computation

$$BR^{-1}B^T \varphi = \chi_{s,t}$$

Solving a **linear system**

Bad news: Solving a linear system can take $\mathbf{O(n^{\omega})=O(n^{2.373})}$

(Prohibitive!)

Key observation:

$\mathbf{BR^{-1}B^T}$ is the **Laplacian** matrix **L**

of the underlying graph

How to compute an electrical flow?

Bot

Laplacian = key object of spectral graph theory (will get back to this)

Electrical flow computation

$$L \varphi = \chi_{s,t}$$

Solving a **Laplacian system**

Bad news: Solving a linear system can take $\mathbf{O(n^{\omega})=O(n^{2.373})}$

(Prohibitive!)

Key observation:

$\mathbf{BR^{-1}B^{T}}$ is the **Laplacian** matrix **L** of the underlying graph

How to compute an electrical flow?

Bottom line:

Electrical flow computation

$$L \varphi = \chi_{s,t}$$

Solving a **Laplacian system**

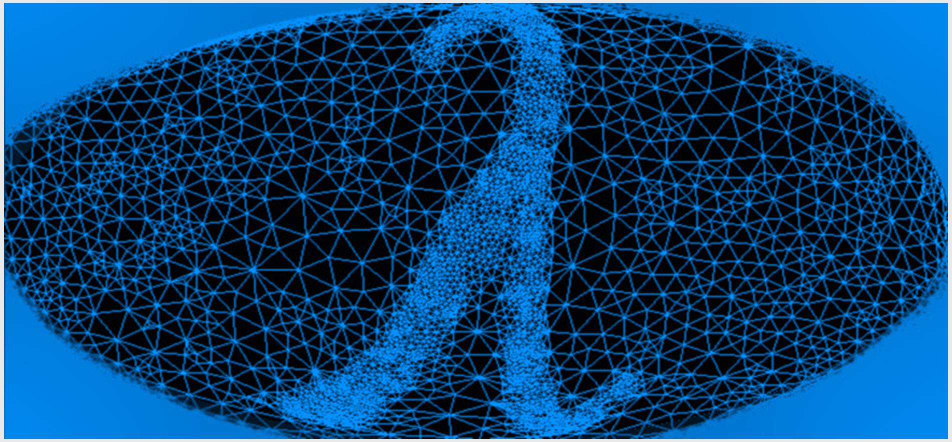
Bad news: Solving a linear system can take $\mathbf{O(n^{\omega})=O(n^{2.373})}$

(Prohibitive!)

Key observation:
$\mathbf{BR^{-1}B^{T}}$ is the **Laplacian** matrix **L** of the underlying graph

(will get back to this)

Result: Electrical flow is a **nearly-linear time** primitive



Addendum: A Glimpse of Spectral Graph Theory

Spectral graph theory: Understanding graphs via eigenvalues and eigenvectors of associated matrices

Central object: **Laplacian** matrix of a graph **G=(V,E,w)**

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

Diagonal **degree** matrix

$\mathbf{D_{vv} = deg(v) = \Sigma_u\, w_{uv}}$

Adjacency matrix

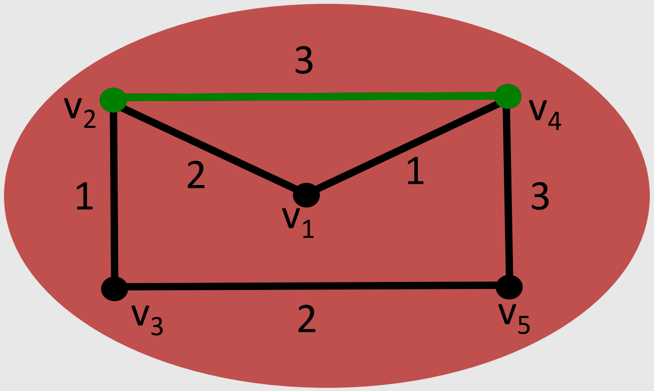
$\mathbf{A_{uv} = w_{uv}}$

Equivalently:

$$\mathbf{L_{uv}} = \begin{cases} \mathbf{-w_{uv}} & \text{if } \mathbf{(u,v)} \text{ in } \mathbf{E} \\ \mathbf{deg(v)} & \text{if } \mathbf{u=v} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

Spectral graph theory: Understanding graphs via eigenvalues and eigenvectors of associated matrices

Example:



$$\mathbf{L} = \begin{bmatrix} 3 & -2 & 0 & -1 & 0 \\ -2 & \boxed{6} & -1 & \boxed{-3} & 0 \\ 0 & -1 & 3 & 0 & -2 \\ -1 & \boxed{-3} & 0 & \boxed{7} & -3 \\ 0 & 0 & -2 & -3 & 5 \end{bmatrix}$$

Observe:

$$\mathbf{L = \Sigma_e\ w_e\ L^e}$$

Laplacian of a graph **(V, {e})**

Laplacian as a quadratic form:

$$\mathbf{x^T\ L\ x = \Sigma_e\ w_e\ x^T\ L^e\ x = \Sigma_e\ w_e\ (x_u - x_v)^2}$$

Spectrum of a Laplacian

Laplacian is an **$n \times n$ symmetric** matrix.

→ It has as **n real eigenvalues $\lambda_1 \le \lambda_2 \le \ldots \le \lambda_n$** with corresponding (orthogonal) **eigenvectors $v^1, v^2, \ldots, v^n$** s.t.

$$L v^i = \lambda_i v^i$$

Can show: $\lambda_1 = 0$ and $v^1 = (1,\ldots,1)$

These objects tell us a lot about the graph!

(And can compute some of λ_is and v^is in **nearly-linear time**)

Most important eigenvalue: λ_2

λ_2 and graph connectivity

Fact: λ_2**=0** iff **G** is disconnected

(**More generally:** λ_k**=0** iff **G** has at least **k** connected components)

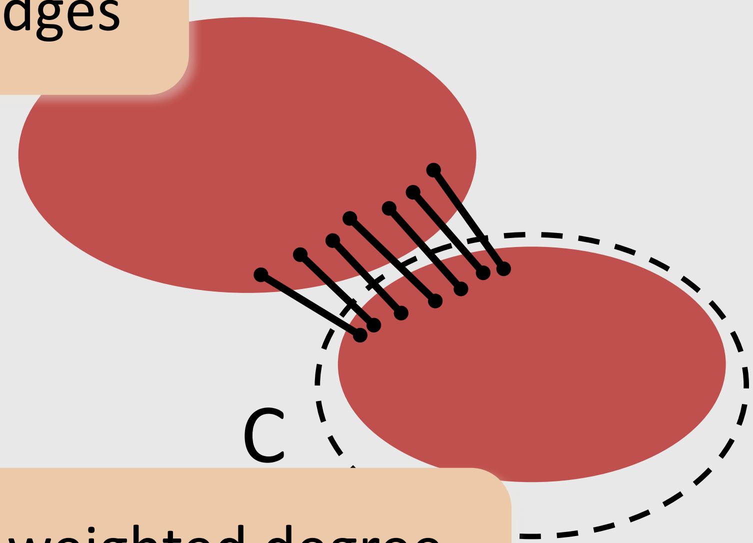
Can we make this connect … e?

Cut conductance:

$$\Phi(C) = \frac{w(C)}{\deg(C)}$$

Total weight of cut edges

Total weighted degree (of the "smaller" side)



λ_2 and graph connectivity

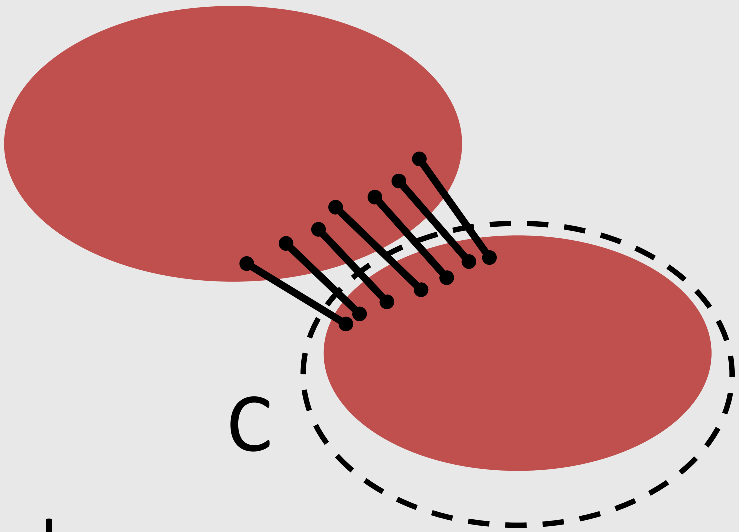
Fact: $\boldsymbol{\lambda_2}$**=0** iff **G** is disconnected

(**More generally:** $\boldsymbol{\lambda_k}$**=0** iff **G** has at least **k** connected components)

Can we make this connection quantitative?

Graph conductance:

$$\Phi_G = \min_C \frac{\mathbf{w(C)}}{\mathbf{deg(C)}}$$



Φ_G **large** → **G** is well connected

Φ_G **small** → **G** has a "bottlenecking" cut

λ_2 and graph connectivity

For a **normalized** Laplacian $\mathcal{L} = \mathbf{D}^{-1/2}\,\mathbf{L}\,\mathbf{D}^{-1/2}$

$$\boldsymbol{\lambda_2/2 \le \Phi_G \le 2\,\lambda_2^{1/2}}$$

[Cheeger '70, Alon-Milman '85]

λ_2 and graph connectivity

For a **normalized** Laplacian $\mathcal{L} = \mathbf{D^{-½} L D^{-½}}$

$$\boldsymbol{\lambda_2/2 \leq \Phi_G \leq 2\,\lambda_2^{½}}$$

[Cheeger '70, Alon-Milman '85]

A cut **C** with $\boldsymbol{\Phi(C) \leq 2\,\lambda_2^{½}}$ can be found in **nearly-linear time**

→ Gives an $\mathbf{O(\lambda_2^{-½})}$**-approx.** to $\boldsymbol{\Phi_G}$
(Computing $\boldsymbol{\Phi_G}$ is **NP-hard**)

→ Great when $\boldsymbol{\lambda_2}$ is large, i.e., **G** is well-connected,
but pretty poor for small $\boldsymbol{\lambda_2}$

λ_2 and graph connectivity

For a **normalized** Laplacian $\mathcal{L} = \mathbf{D^{-1/2} L D^{-1/2}}$

$$\lambda_2/2 \le \Phi_G \le 2\,\lambda_2^{1/2}$$

[Cheeger '70, Alon-Milman '85]

A cut **C** with $\mathbf{\Phi(C) \le 2\,\lambda_2^{1/2}}$ can be found in **nearly-linear time**

→ Gives an $\mathbf{O(\lambda_2^{-1/2})}$**-approx.** to $\mathbf{\Phi_G}$
(Computing $\mathbf{\Phi_G}$ is **NP-hard**)

→ Great when $\boldsymbol{\lambda_2}$ is large, i.e., **G** is well-connected,
but pretty poor for small $\boldsymbol{\lambda_2}$

Unfortunately: The $\mathbf{\lambda_2^{1/2}}$ **vs.** $\boldsymbol{\lambda_2}$ gap is unavoidable

λ_2 and random walks

Paint spilling process:

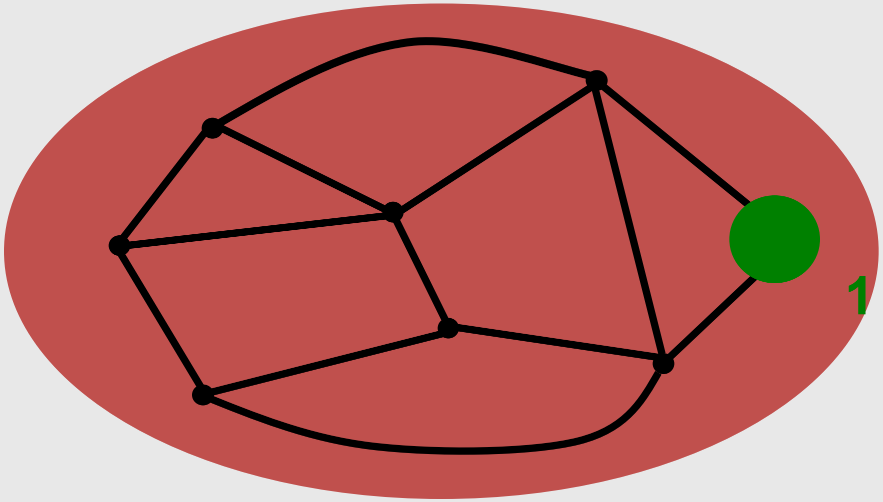
→ **Start with all paint at s**

→ **For each vertex:**

Split the paint in half:

– one half stays put

– distribute the rest (evenly) among the neighbors



λ_2 and random walks

Paint spilling process:

→ **Start with all paint at s**

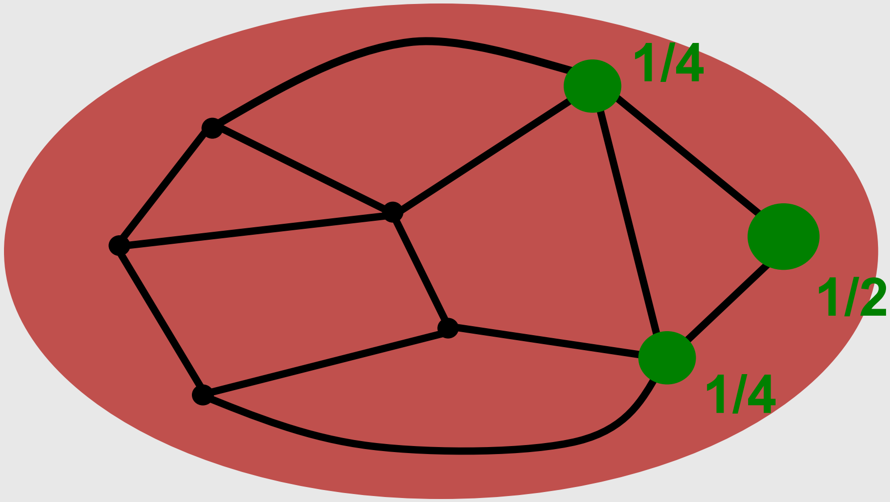
→ **For each vertex:**

Split the paint in half:

– one half stays put

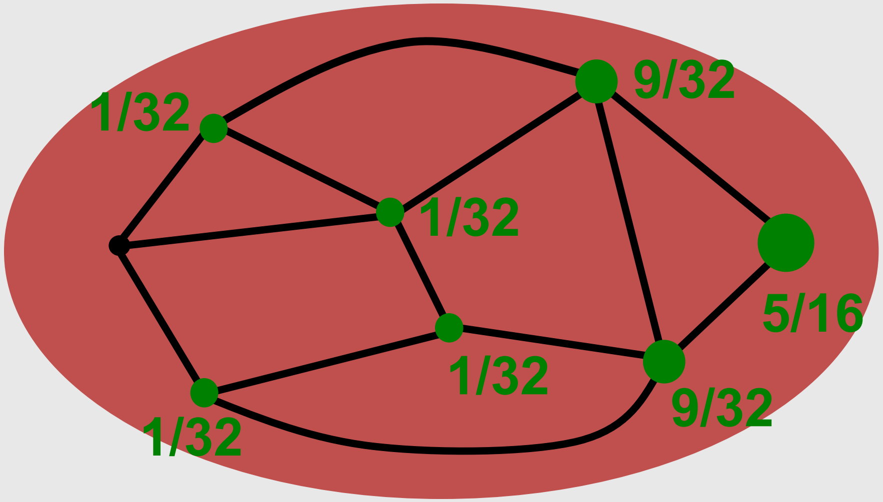
– distribute the rest (evenly) among the neighbors

→ **Repeat**



λ_2 and random walks

Paint spilling process:
→ **Start with all paint at s**
→ **For each vertex:**
Split the paint in half:
– one half stays put
– distribute the rest (evenly) among the neighbors
→ **Repeat**



This diffusive process corresponds to a **(lazy) random walk** and shows up everywhere

Fact: Paint distribution **always*** converges to a **stationary distribution π** with $\pi_v \sim$ **deg(v)**

But: How fast is this convergence?

Theorem: The convergence rate is $\Theta(\lambda_2^{-1})$

Beyond λ_2?

→ Looking at the higher order eigenvalues

For any $\mathbf{k \geq 2}$,

$$\Phi_G \leq O(k)\, \lambda_2/\lambda_k^{1/2}$$ **[LOT'12, KLLOT'13]**

→ **Electrical graph theory:** Using electrical flows

Key quantity: Effective resistance (between **s** and **t**)

$$R_{st} = \chi_{st}^T L^+ \chi_{st}$$

Vector with **1** at **t**, **-1** at **s** and **0**s everywhere else

Pseudo-inverse of the Laplacian

Beyond λ_2?

→ Looking at the higher order eigenvalues

For any **$k \geq 2$**,

$$\Phi_G \leq O(k)\ \lambda_2/\lambda_k^{1/2}$$ **[LOT'12, KLLOT'13]**

→ **Electrical graph theory:** Using electrical flows

Key quantity: Effective resistance (between **s** and **t**)

$$R_{st} = \chi_{st}^T L^+ \chi_{st}$$

Note: Effective resistance depends on the **whole** spectrum of **L**

[SS '08]: We can (approx.) compute **all** resistances in **nearly-linear time**

Electrical flows show up in many contexts:

- → Behavior of random walks (commute time, PageRank,…)
- → Graph sparsification
- → Sampling random spanning trees
- → **Maximum flow problem**

Where else can we use them?

Thank you

Next: (Sub)Gradient Descent Method