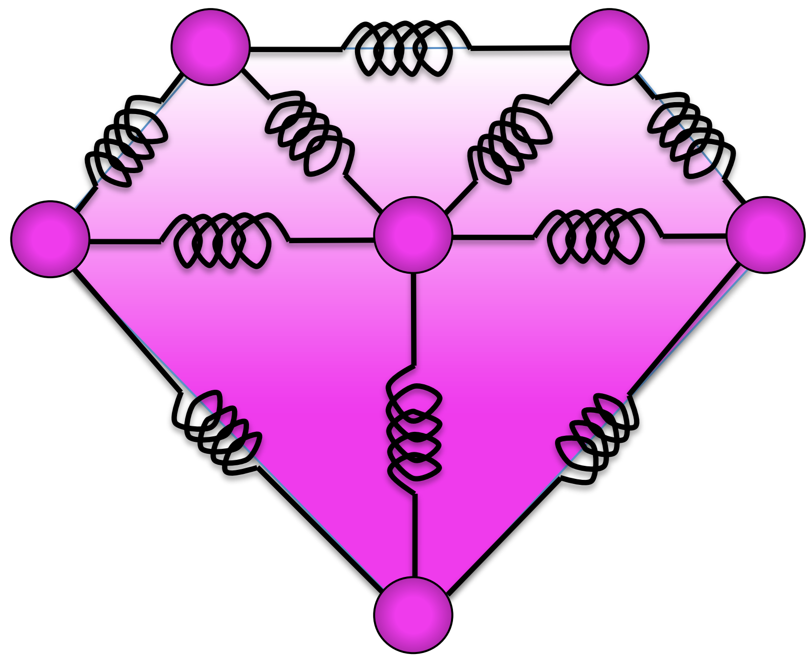
Laplacian Gems



Daniel A. Spielman
Yale University

Outline

Laplacian Linear Systems (Electrical Graph Theory)

Applications

Fast Algorithms

- Bold ideas
- Surprising results
- Powerful techniques

Faster than Koutis-Miller-Peng?

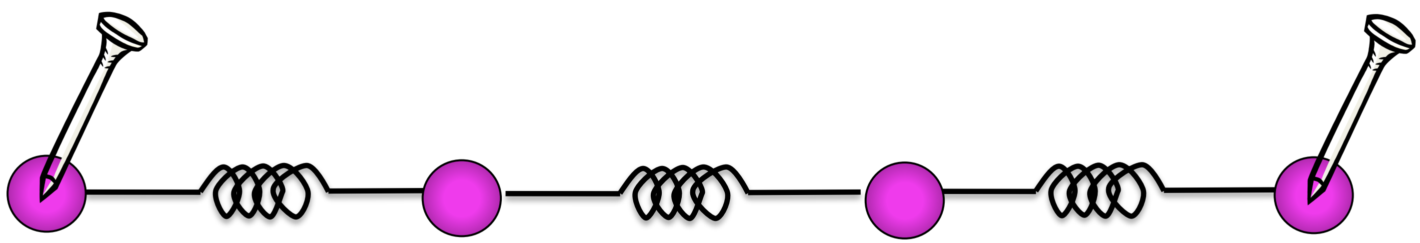
Graphs as Spring Networks

edges -> ideal linear springs
weights -> spring constants (k)

Physics: when stretched to length x, force is kx

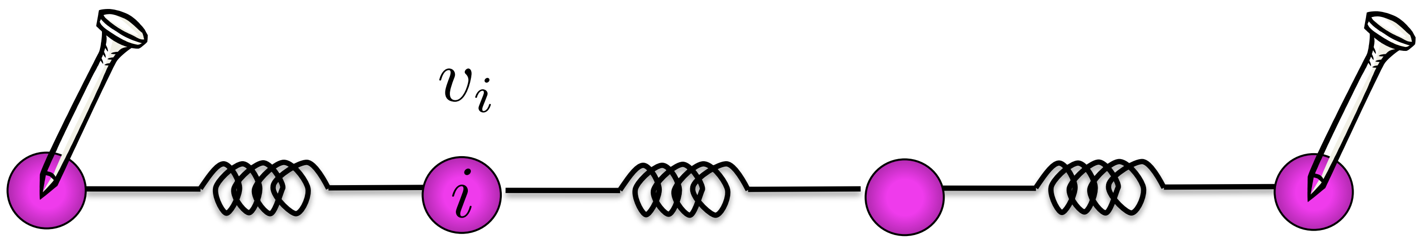
potential energy is $kx^2/2$

Nail down some vertices, let rest settle



Graphs as Spring Networks

Nail down some vertices, let rest settle



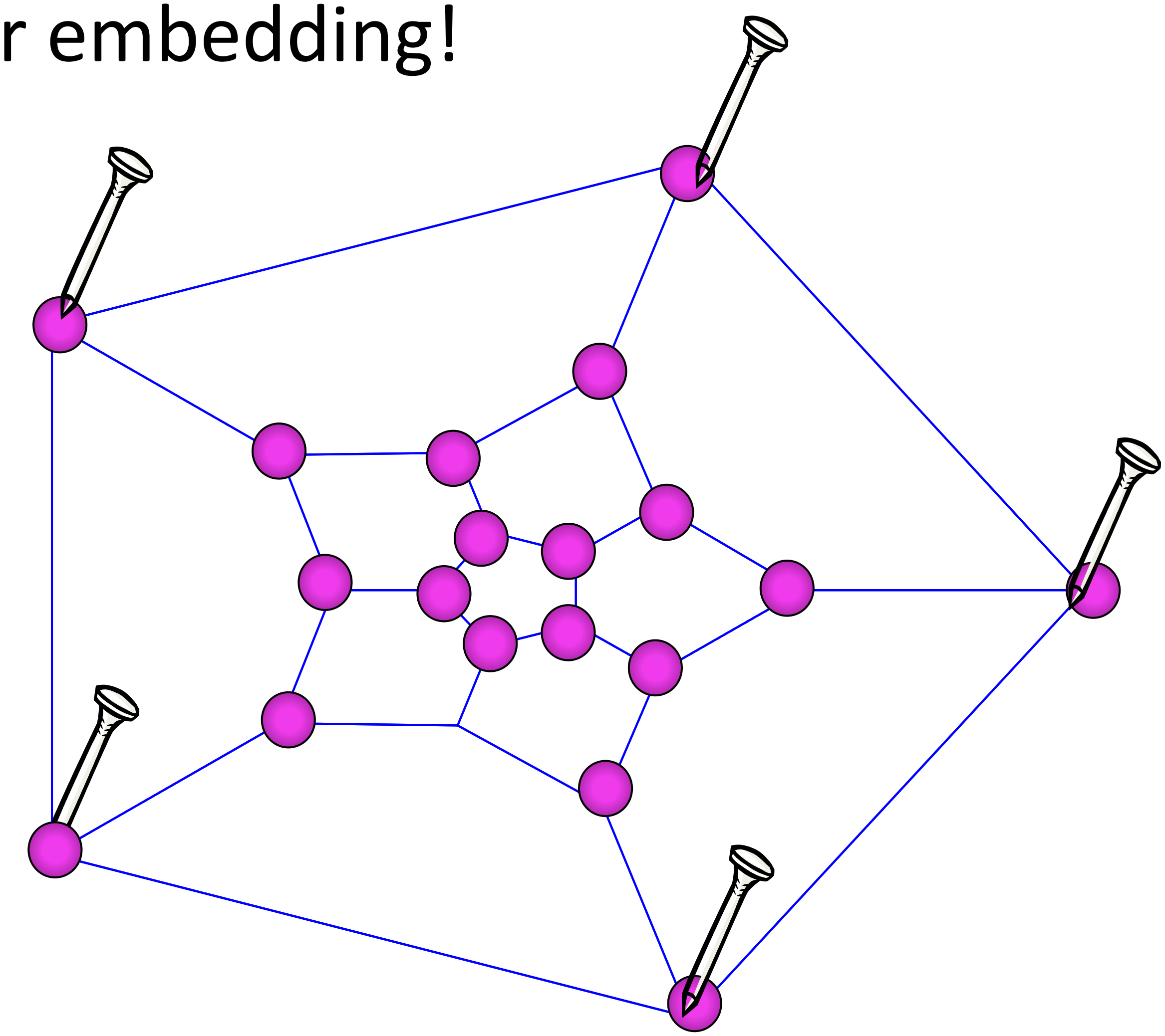
Physics: minimizes total potential energy

$$\sum_{(i,j)\in E} w_{i,j}(v_i - v_j)^2 = v^T L v$$

subject to boundary constraints (nails)

Tutte's Theorem '63

If nail down a face of a planar 3-connected graph, get a planar embedding!



The Laplacian

$$v^T L v = \sum_{(i,j)\in E} w_{i,j}(v_i - v_j)^2$$

$$L_{i,j} = \begin{cases} -w_{i,j} & \text{if } (i,j) \in E \\ d_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \qquad d_i = \sum_j w_{i,j}$$

$$L = \sum_{(i,j)\in E} w_{i,j}(e_i - e_j)(e_i - e_j)^T$$

The Laplacian

$$v^T L v = \sum_{(i,j) \in E} w_{i,j} (v_i - v_j)^2$$

$$L_{i,j} = \begin{cases} -w_{i,j} & \text{if } (i,j) \in E \\ d_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \qquad d_i = \sum_j w_{i,j}$$

Can minimize, subject to boundary $v_i = f_i \text{ for } i \in S$

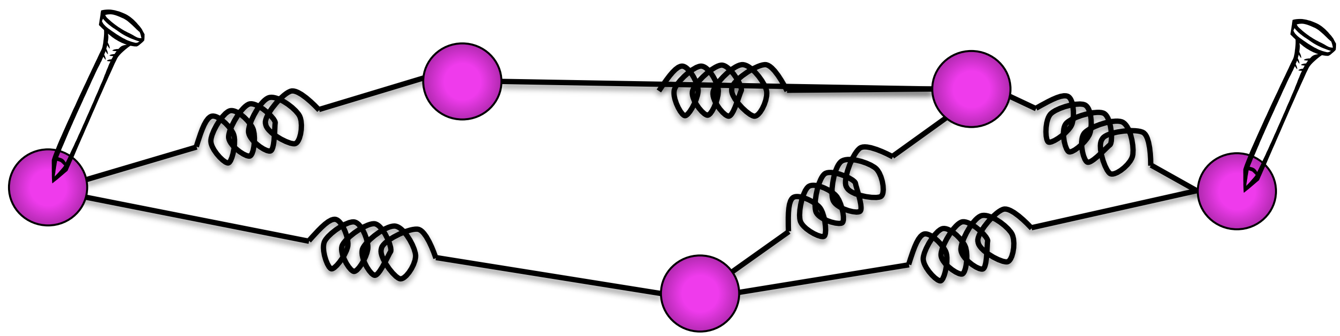
by Laplacian Linear Solvers in nearly-linear time

[Koutis-Miller-Peng '10, S-Teng '04]

Regression on Graphs [Zhu-Ghahramani-Lafferty '03]

Say know f(i) for all i in S $\subset$ V
want to guess f(j) for all other j.

Idea: nail node i to f(i) on Real line.



$$\min \sum_{(i,j)\in E} w_{i,j}(v_i - v_j)^2 = v^T L v$$

Graphs as Resistor Networks

edge -> resistor
weight -> 1/resistance

Ohm's law: flow/current

$$f_{i,j} = \frac{v_i - v_j}{r_{i,j}} = w_{i,j}(v_i - v_j)$$

Graphs as Resistor Networks

edge -> resistor
weight -> 1/resistance

Ohm's law: flow/current

$$f_{i,j} = \frac{v_i - v_j}{r_{i,j}} = w_{i,j}(v_i - v_j)$$

Demands/external current:

$$d = Lv \qquad\qquad v = L^+ d$$

the pseudo-inverse

Graphs as Resistor Networks

Ohm's law: flow/current

$$f_{i,j} = \frac{v_i - v_j}{r_{i,j}} = w_{i,j}(v_i - v_j)$$

Demands/external current:

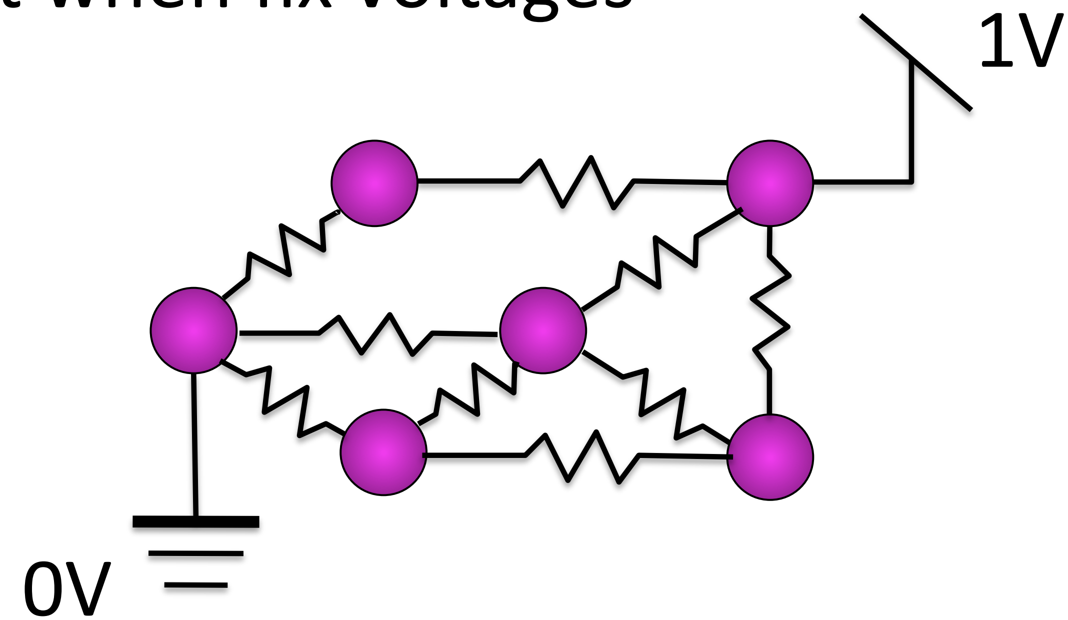
$$d = Lv \qquad\qquad v = L^+ d$$

To flow 1 from s to t:

- Set d(s) = 1 and d(t) = -1
- Solve for v
- Use Ohm's law to find flow in graph

Graphs as Resistor Networks

To solve for current when fix voltages



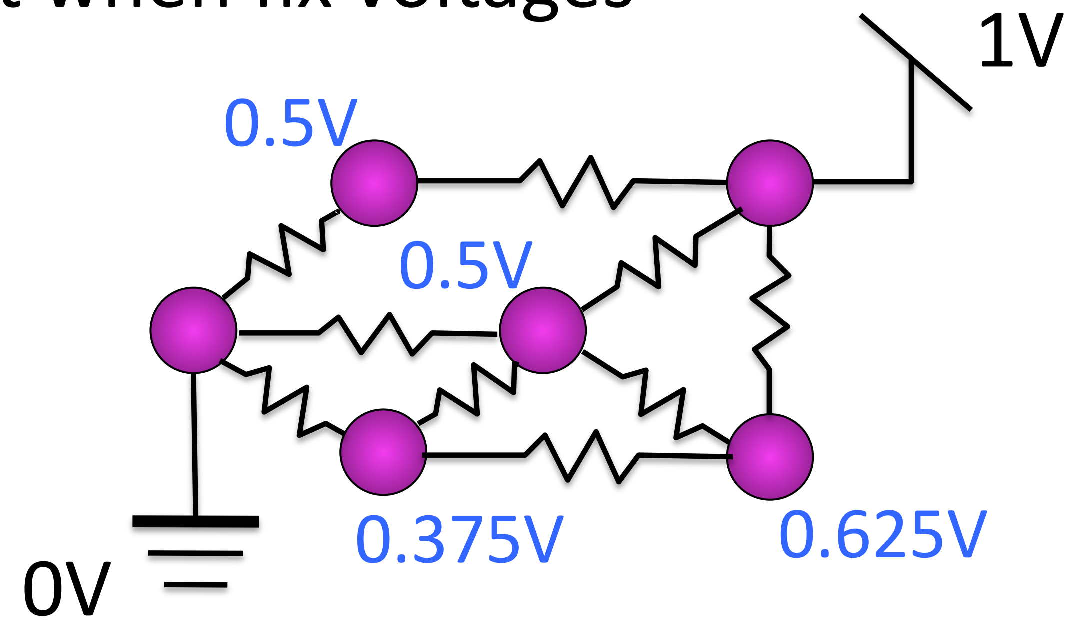
Solve for v minimizing dissipated energy,

$$v^T L v = \sum w_{i,j}(v_i - v_j)^2$$

subject to fixed potentials

Graphs as Resistor Networks

To solve for current when fix voltages

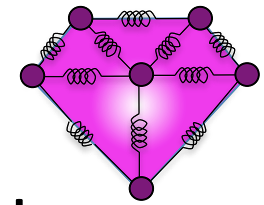


Solve for v minimizing dissipated energy,

$$v^T L v = \sum w_{i,j} (v_i - v_j)^2$$

subject to fixed potentials

Effective Resistance between s and t



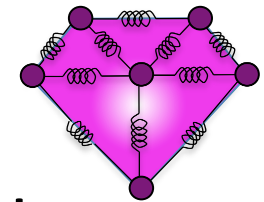
= resistance of whole network between s and t

= $v_s - v_t$ in s-t flow of value 1

$$= (e_s - e_t)^T L^+ (e_s - e_t)$$

$$f_{i,j} = \frac{v_i - v_j}{r_{i,j}}$$

Effective Resistance between s and t



= resistance of whole network between s and t

= $v_s - v_t$ in s-t flow of value 1

$$= (e_s - e_t)^T L^+ (e_s - e_t)$$

= 1/effective spring constant between s and t

= prob edge (s,t) in a random spanning tree

~ expected commute time between s and t
[Chandra-Raghavan-Ruzzo-Smolensky-Tiwari '89]

And, is a distance.

Application: Random Spanning Trees

Can use fast Laplacian solvers to sample random spanning trees in time $\tilde{O}(mn^{1/2})$ [Kelner-Madry-Propp '09]

Idea: accelerate random-walk based sampling [Aldous, Broder] by jumping through clusters where walk is slow.

Application: Approximate Maximum Flow

[Christiano-Kelner-Madry-S-Teng '10]

1. Compute electrical s-t flow
2. Increase resistance on over-capacity edges
3. Return average of computed flows (a la MWM)

Time $\tilde{O}(m^{3/2}/\epsilon^3)$

Application: Approximate Maximum Flow

[Christiano-Kelner-Madry-S-Teng '10]

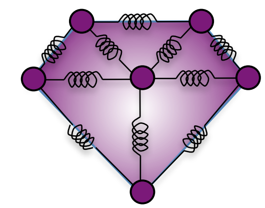
1. Compute electrical s-t flow
2. Increase resistance on over-capacity edges
   and delete very over-capacity edges
3. Return average of computed flows (a la MWM)

Time $\tilde{O}(m^{3/2}/\epsilon^3) \longrightarrow \tilde{O}(mn^{1/3}/\epsilon^{11/3})$

Approximate Laplacian Solvers

Preconditioned Conjugate Gradient [Hestenes '51, Stiefel '52, ???]	$O(mn)$
Vaidya '90: Subgraph preconditioners	$O(mn^{3/4})$
Boman-Hendrickson '01: Using Low-Stretch Spanning Trees	$\tilde{O}(mn^{1/2})$
S-Teng '04: Spectral sparsification	$\tilde{O}(m)$
Koutis-Miller-Peng '10: Elegance	$O(m \log^2 n)$

Preconditioned Conjugate Gradient



Solve $L_G x = b$ by

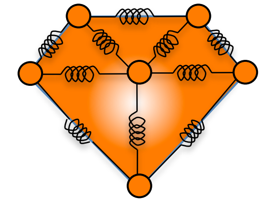
Approximating L_G by L_H (the preconditioner)

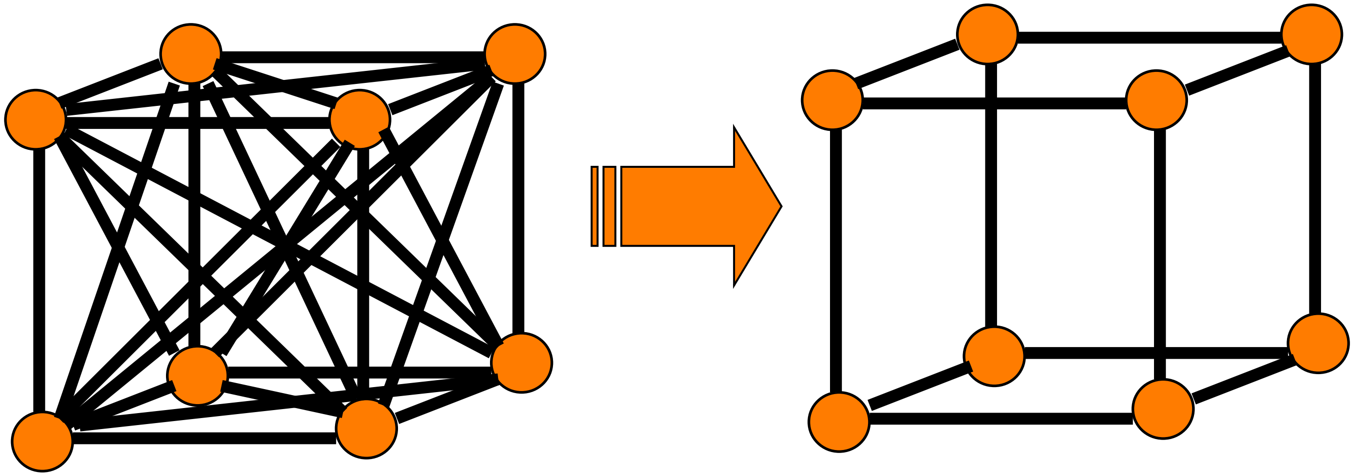
In each iteration
solve a system in L_H
multiply a vector by L_G

ϵ-approx solution after
$O(\sqrt{\kappa(L_G, L_H)} \log \epsilon^{-1})$ iterations

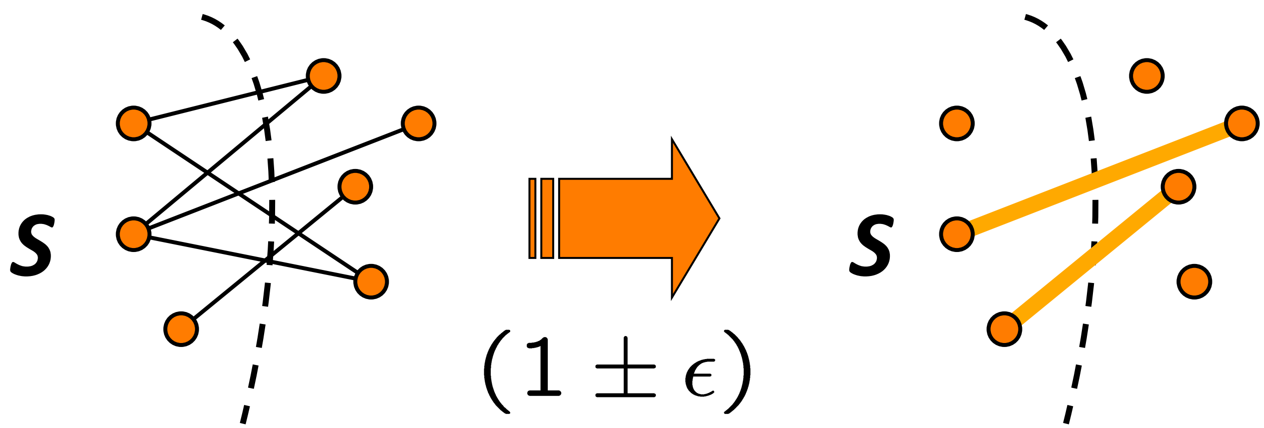
condition number/approx quality

Graph Sparsification [Benczur-Karger '96]

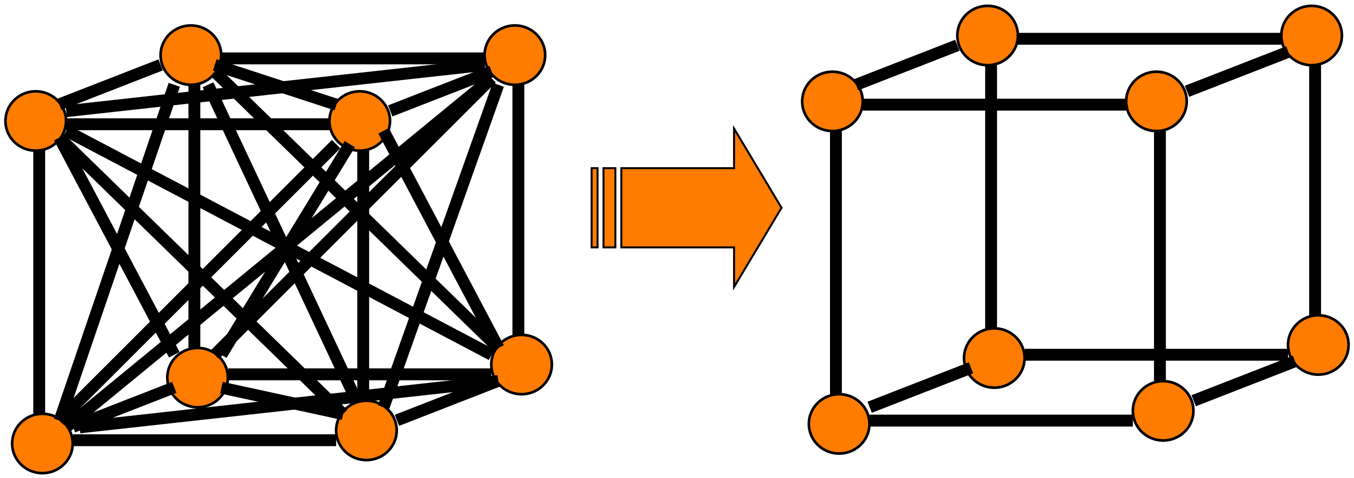




Approximate G by a sparse H,
approximately preserving all cuts



Spectral Sparsification [S-Teng '04]



Approximate G by a sparse H,
approximately preserving all energies
and effective resistances

$$v^T L_G v \quad \xrightarrow[(1 \pm \epsilon)]{} \quad v^T L_H v$$

Spectral Sparsification [S-Srivastava '08]

$$v^T L_G v \quad \underset{(1 \pm \epsilon)}{\Longrightarrow} \quad v^T L_H v$$

Sample edges (i,j) of G
with probability proportional to $R_{eff}(i,j) w_{i,j}$

$O(n \log n / \epsilon^2)$ edges suffice

Analysis by random matrix concentration results
of Rudelson and Vershynin.

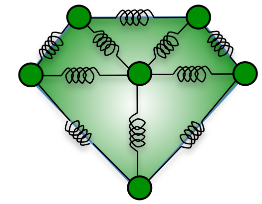
Spectral Sparsification [Koutis-Miller-Peng '10]

Understand Rudelson-Vershynin much better

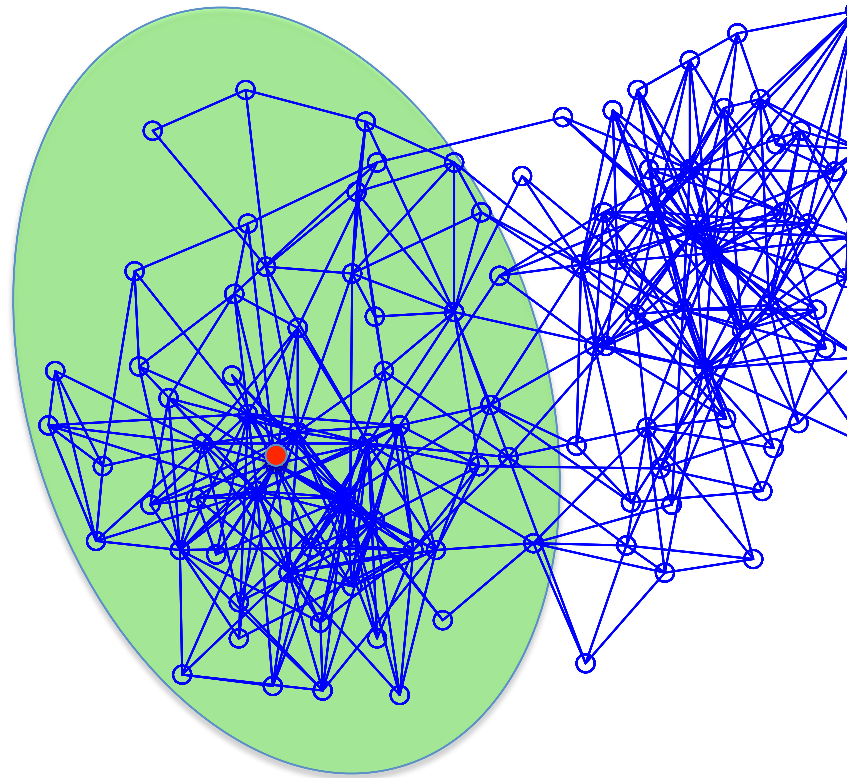
Suffices to sample edges (i,j) of G
with probability crudely related to $R_{eff}(i,j)w_{i,j}$

Sample $O(n \log n/\epsilon^2)$ edges ***with replacement***
so actually get many fewer edges!

Local Graph Clustering [S-Teng '04]

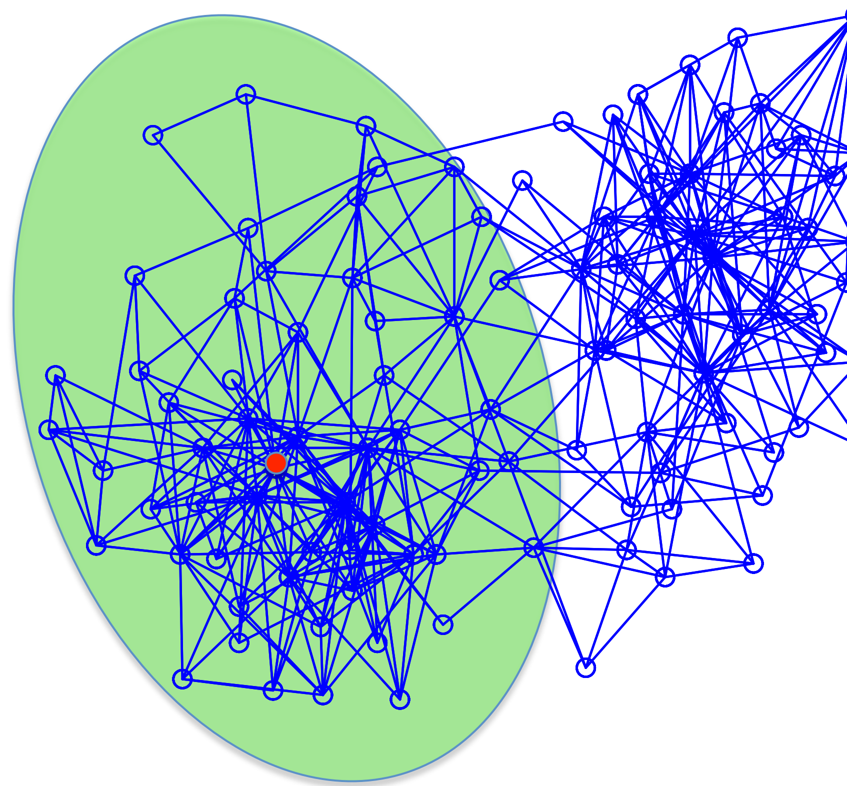


Given vertex of interest find nearby cluster S with small conductance in time O(|S|)



Local Graph Clustering [S-Teng '04]

Given vertex of interest
find nearby cluster S
with small conductance
in time O(|S|)



How should one explore a graph?

How should one explore a graph?

Possible goals:

find nodes like the first node

find a representative sample

minimize computation and graph access

prove something rigorous

How should one explore a graph?

Possible goals:

find nodes like the first node

find a representative sample

minimize computation and graph access

prove something rigorous

Beat BFS

Local Graph Clustering [S-Teng '04]

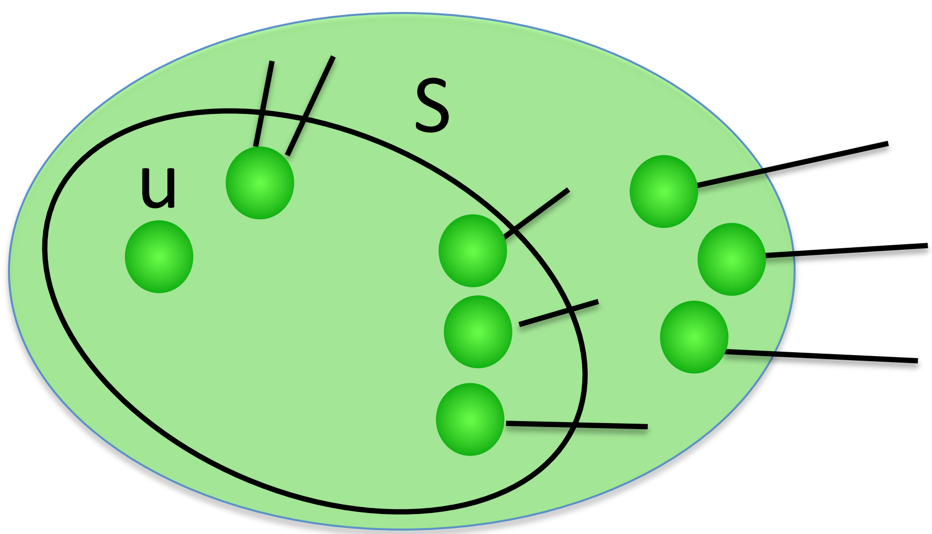
Prove: if S has small conductance ϕ

u is a random node in S

probably

find a set of small conductance, $\phi^{1/2} \log^c n$

in time $|S| \log^c n / \phi^2$



Using Approximate Personal PageRank Vectors

Jeh-Widom '03, Berkhin '06, Andersen-Chung-Lang '06

Spilling paint in a graph:

start at one node

at each step,

α fraction dries

of wet paint, half stays put, half to neighbors

Using Approximate Personal PageRank Vectors

Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

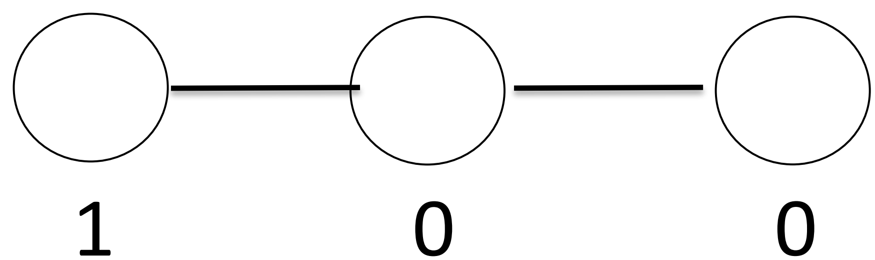
Spilling paint in a graph:

start at one node

at each step,

α fraction dries

of wet paint, half stays put, half to neighbors



Using Approximate Personal PageRank Vectors

Jeh-Widom '03, Berkhin '06, Andersen-Chung-Lang '06

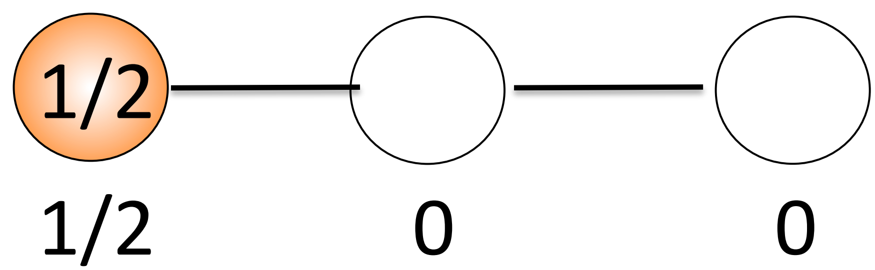
Spilling paint in a graph:

start at one node

at each step,

α fraction dries

of wet paint, half stays put, half to neighbors



with $\alpha = 1/2$

Using Approximate Personal PageRank Vectors

Jeh-Widom '03, Berkhin '06, Andersen-Chung-Lang '06

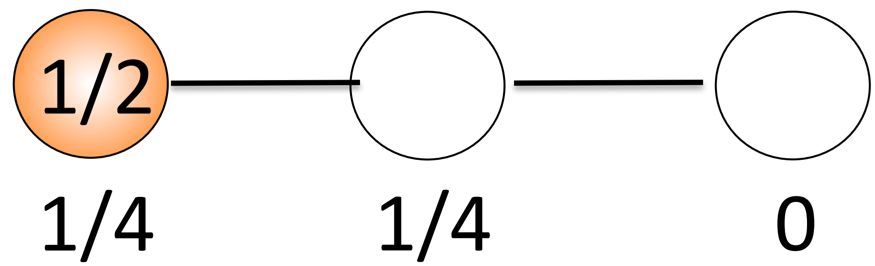
Spilling paint in a graph:

start at one node

at each step,

α fraction dries

of wet paint, half stays put, half to neighbors



with α = 1/2

Using Approximate Personal PageRank Vectors

Jeh-Widom '03, Berkhin '06, Andersen-Chung-Lang '06

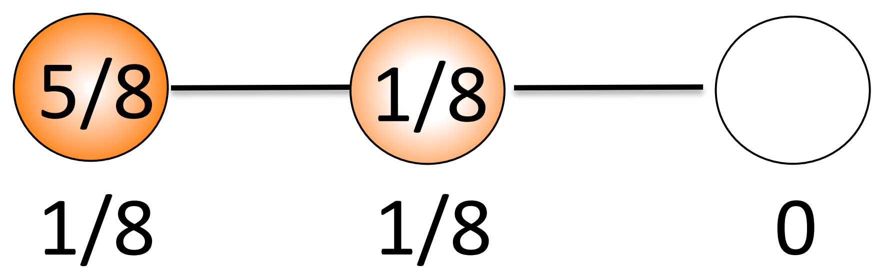
Spilling paint in a graph:

start at one node

at each step,

α fraction dries

of wet paint, half stays put, half to neighbors



with $\alpha = 1/2$

Using Approximate Personal PageRank Vectors

Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

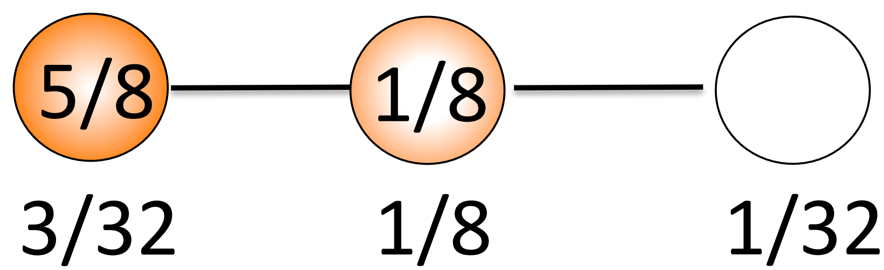
Spilling paint in a graph:

start at one node

at each step,

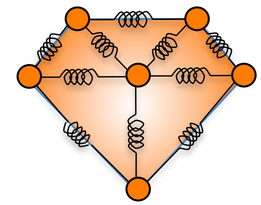
α fraction dries

of wet paint, half stays put, half to neighbors



with $\alpha = 1/2$

Using Approximate Personal PageRank Vectors



Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

Spilling paint in a graph:

  start at one node

  at each step,

    α fraction dries

    of wet paint, half stays put, half to neighbors

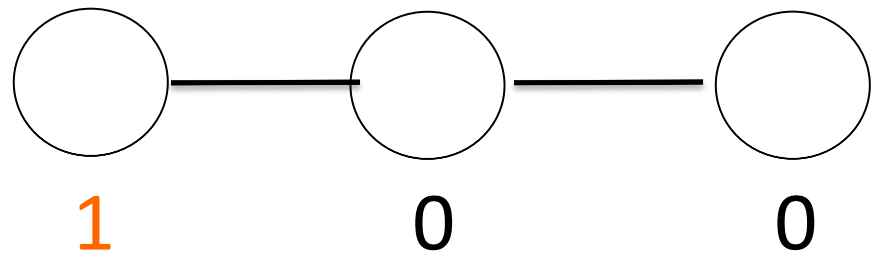
Time doesn’t matter, can push paint whenever

Approximate: only push when a lot of paint

Using Approximate Personal PageRank Vectors

Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

Spilling paint in a graph:
start at one node
at each step,
α fraction dries
of wet paint, half stays put, half to neighbors



Time doesn’t matter, can push paint whenever

Approximate: only push when a lot of paint

Using Approximate Personal PageRank Vectors

Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

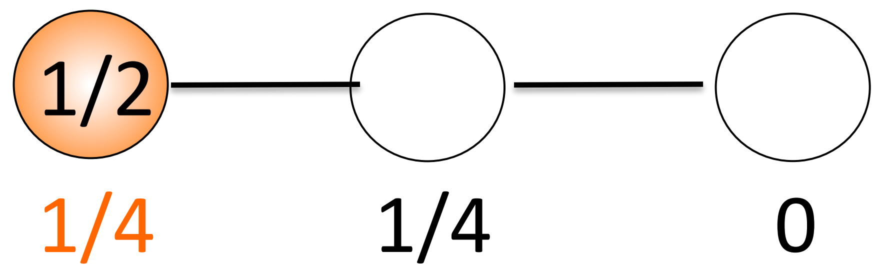
Spilling paint in a graph:

start at one node

at each step,

α fraction dries

of wet paint, half stays put, half to neighbors



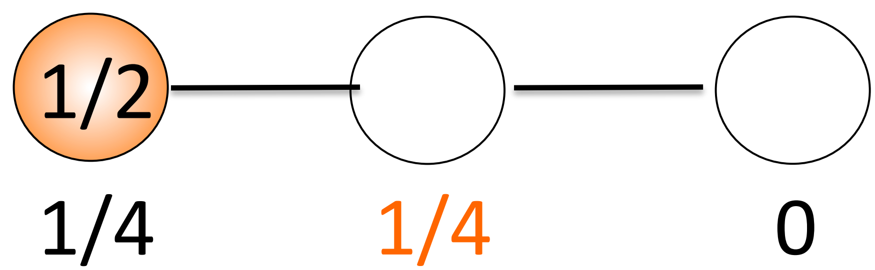
Time doesn’t matter, can push paint whenever

Approximate: only push when a lot of paint

Using Approximate Personal PageRank Vectors

Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

Spilling paint in a graph:
- start at one node
- at each step,
  - α fraction dries
  - of wet paint, half stays put, half to neighbors



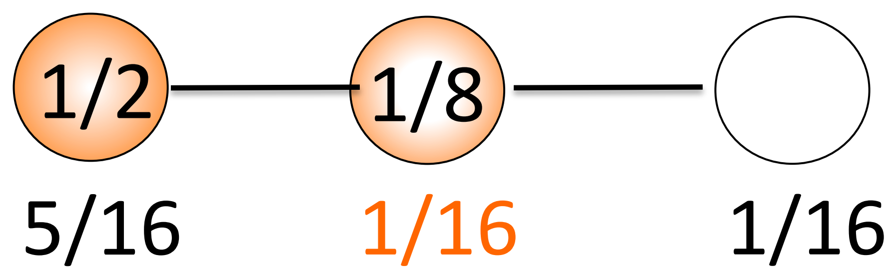
Time doesn’t matter, can push paint whenever

Approximate: only push when a lot of paint

Using Approximate Personal PageRank Vectors

Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

Spilling paint in a graph:
  start at one node
  at each step,
    α fraction dries
    of wet paint, half stays put, half to neighbors



Time doesn’t matter, can push paint whenever

Approximate: only push when a lot of paint

Volume-Biased Evolving Set Markov Chain

[Andersen-Peres '09]

Walk on sets of vertices
starts at one vertex, ends at V

Dual to random walk on graph

When start inside set of conductance ϕ
find set of conductance $\phi^{1/2} \log^{1/2} n$
with work $|S| \log^c n / \phi^{1/2}$

Volume-Biased Evolving Set Markov Chain

[Andersen-Peres '09]

Walk on sets of vertices
 starts at one vertex, ends at V

Dual to random walk on graph

When start inside set of conductance ϕ
 find set of conductance $\phi^{1/2} \log^{1/2} n$
 with work $|S| \log^c n / \underline{\phi^{1/2}}$

can we eliminate this?

A Technique for proving convergence of walks on graphs of Lovász and Simonovits

Consider lazy random walk on a regular graph (stays put with probability ½).

Approaches uniform in time 1/conductance^2.

A Technique for proving convergence of walks on graphs of Lovász and Simonovits

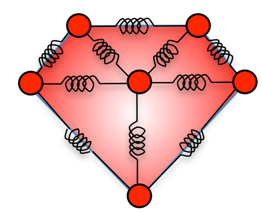
Consider lazy random walk on a regular graph (stays put with probability ½).

Approaches uniform in time 1/conductance^2.

$p^t(i)$ = prob of vertex i at time t

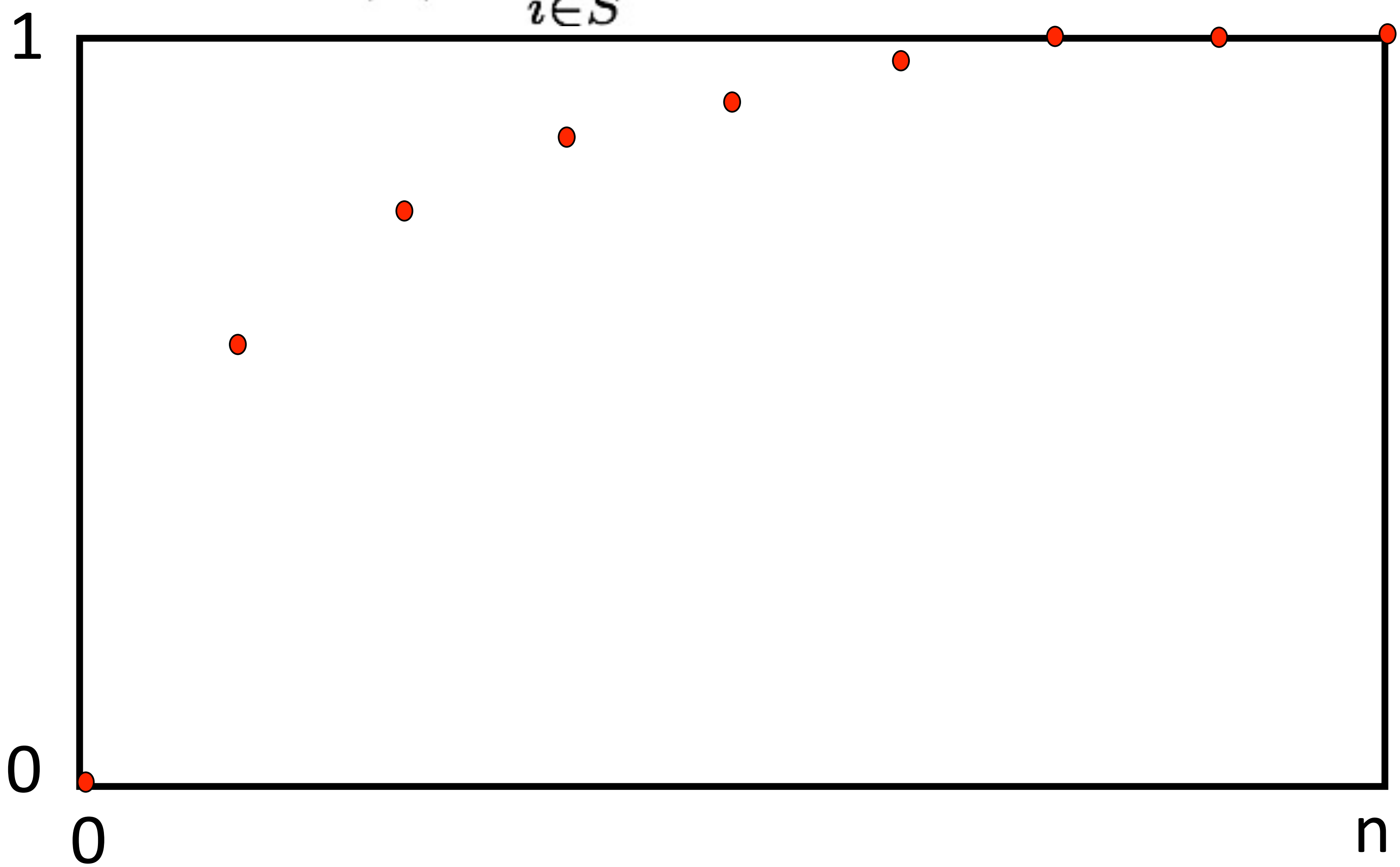
$$C^t(k) = \max_{|S|=k} \sum_{i \in S} p^t(i)$$

sum of largest k values



Lovász-Simonovits

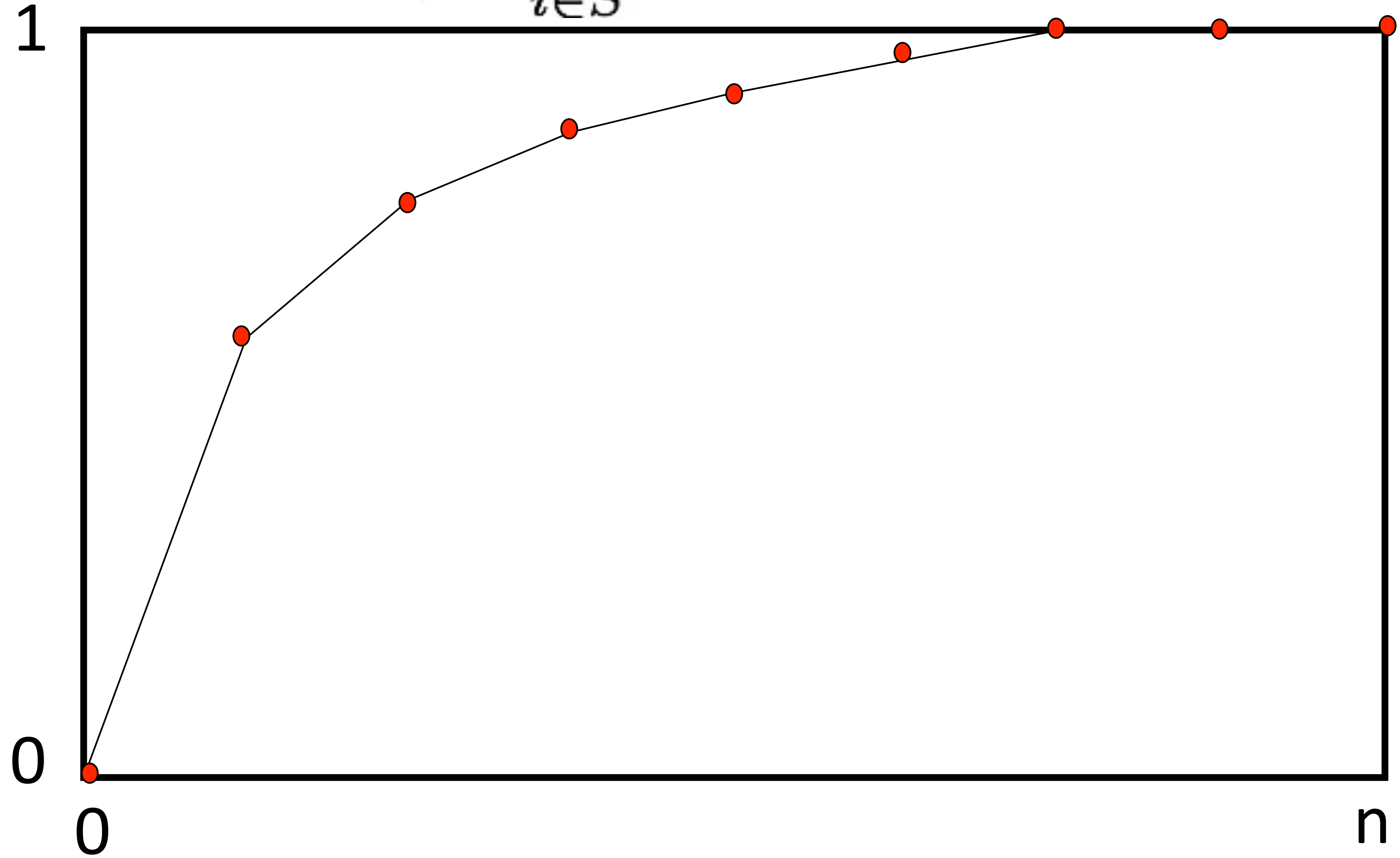
$$C^t(k) = \max_{|S|=k} \sum_{i \in S} p^t(i)$$



Lovász-Simonovits

$$C^t(k) = \max_{|S|=k} \sum_{i \in S} p^t(i)$$

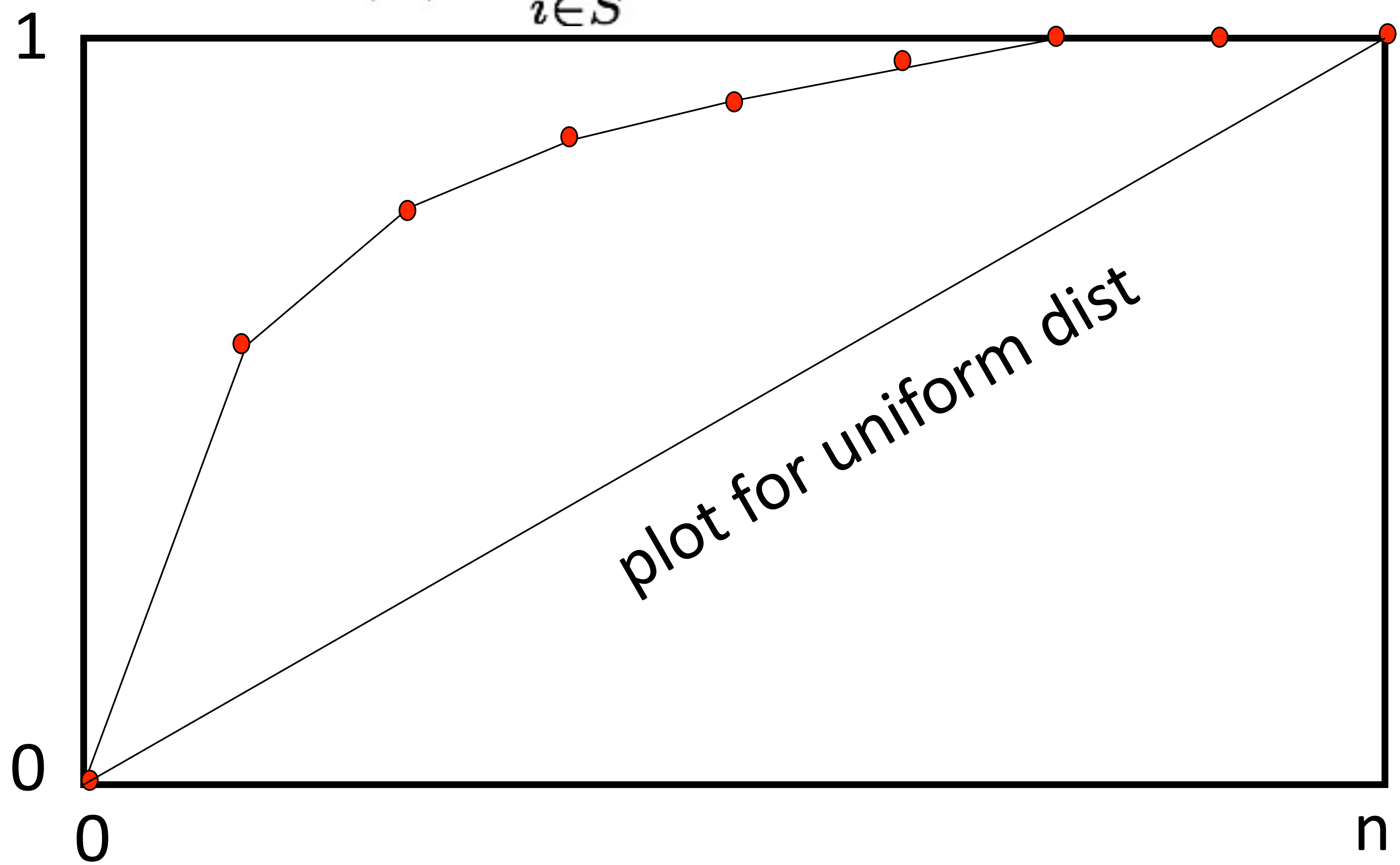
extend by linearity



Lovász-Simonovits

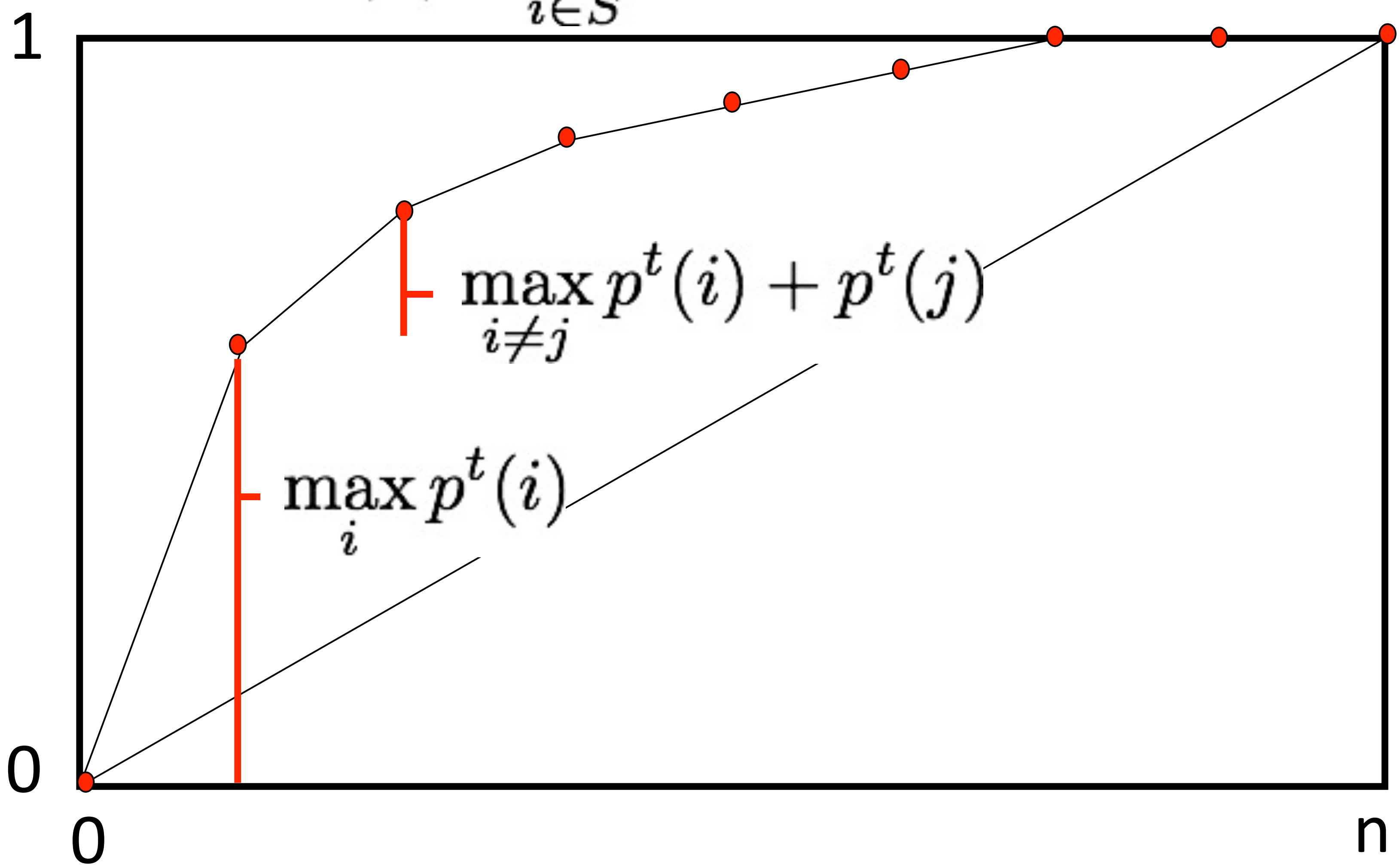
$$C^t(k) = \max_{|S|=k} \sum_{i \in S} p^t(i)$$

interpolate linearly



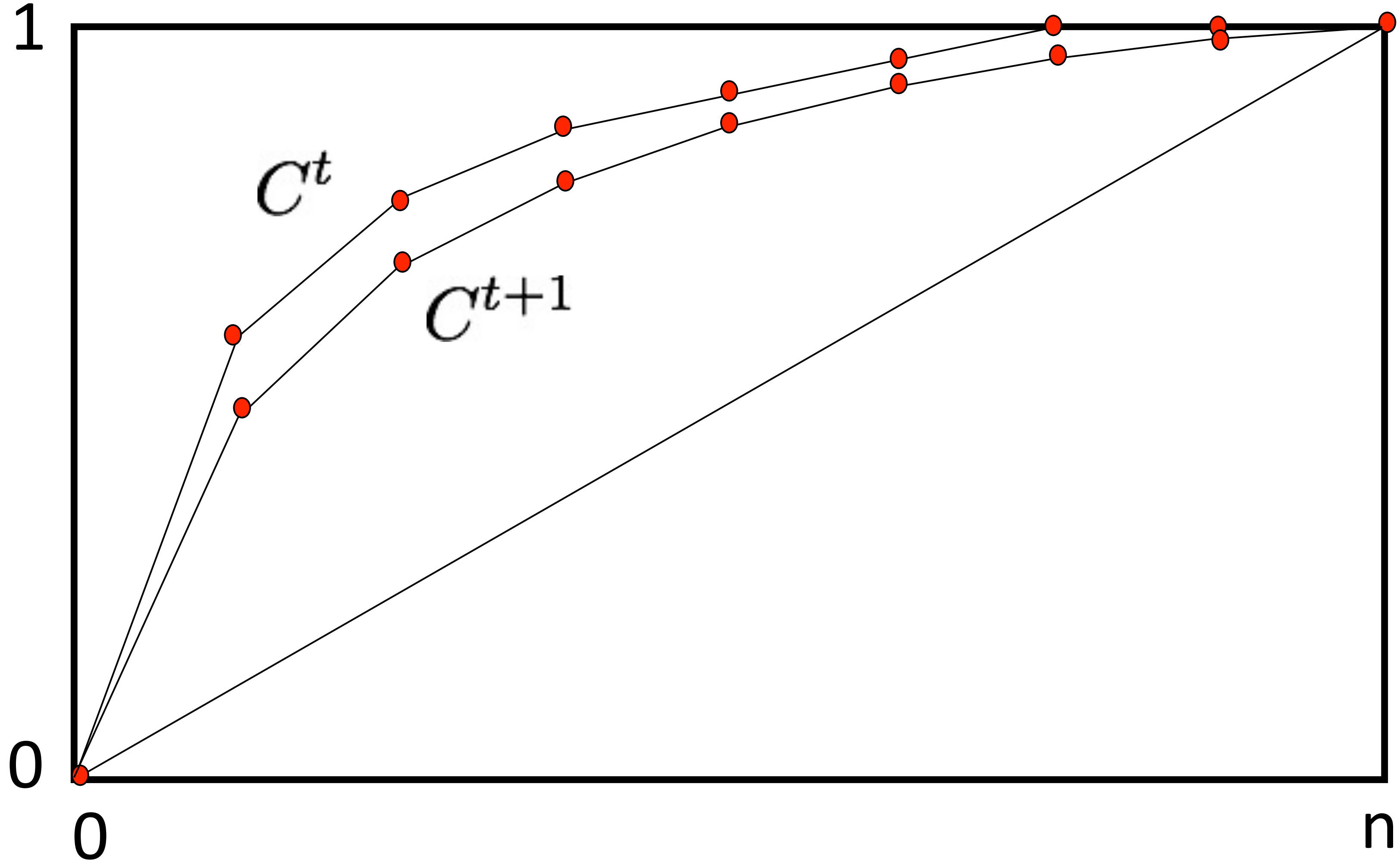
Lovász-Simonovits

$$C^t(k) = \max_{|S|=k} \sum_{i \in S} p^t(i)$$ is concave



Lovász-Simonovits: Easy inequality

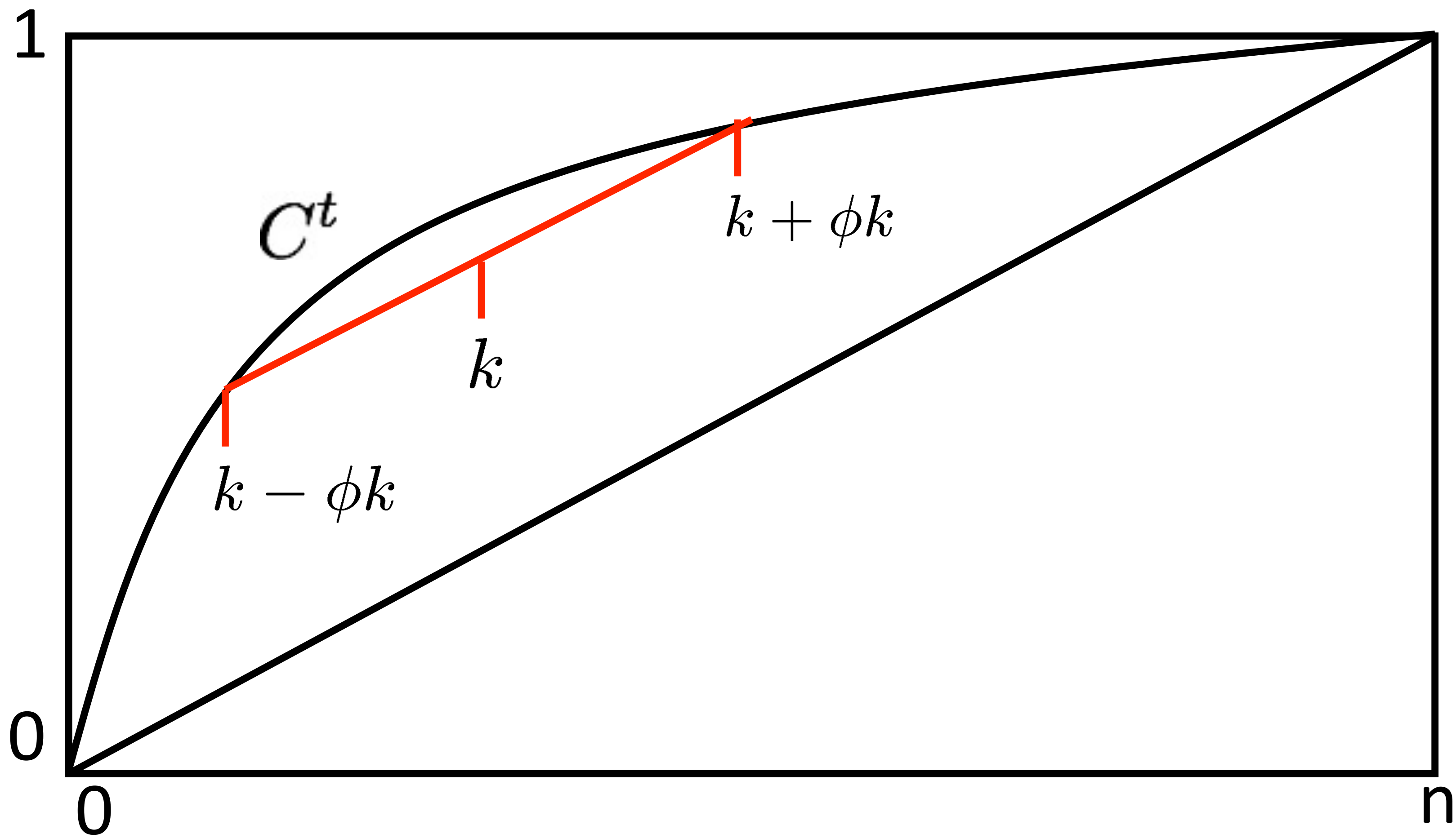
$$C^{t+1} \leq C^t$$



Lovász-Simonovits: Conductance ϕ inequality

$$C^{t+1}(k) \le \frac{1}{2}\left(C^t(k-\phi k) + C^t(k+\phi k)\right) \qquad k \in [0, n/2]$$

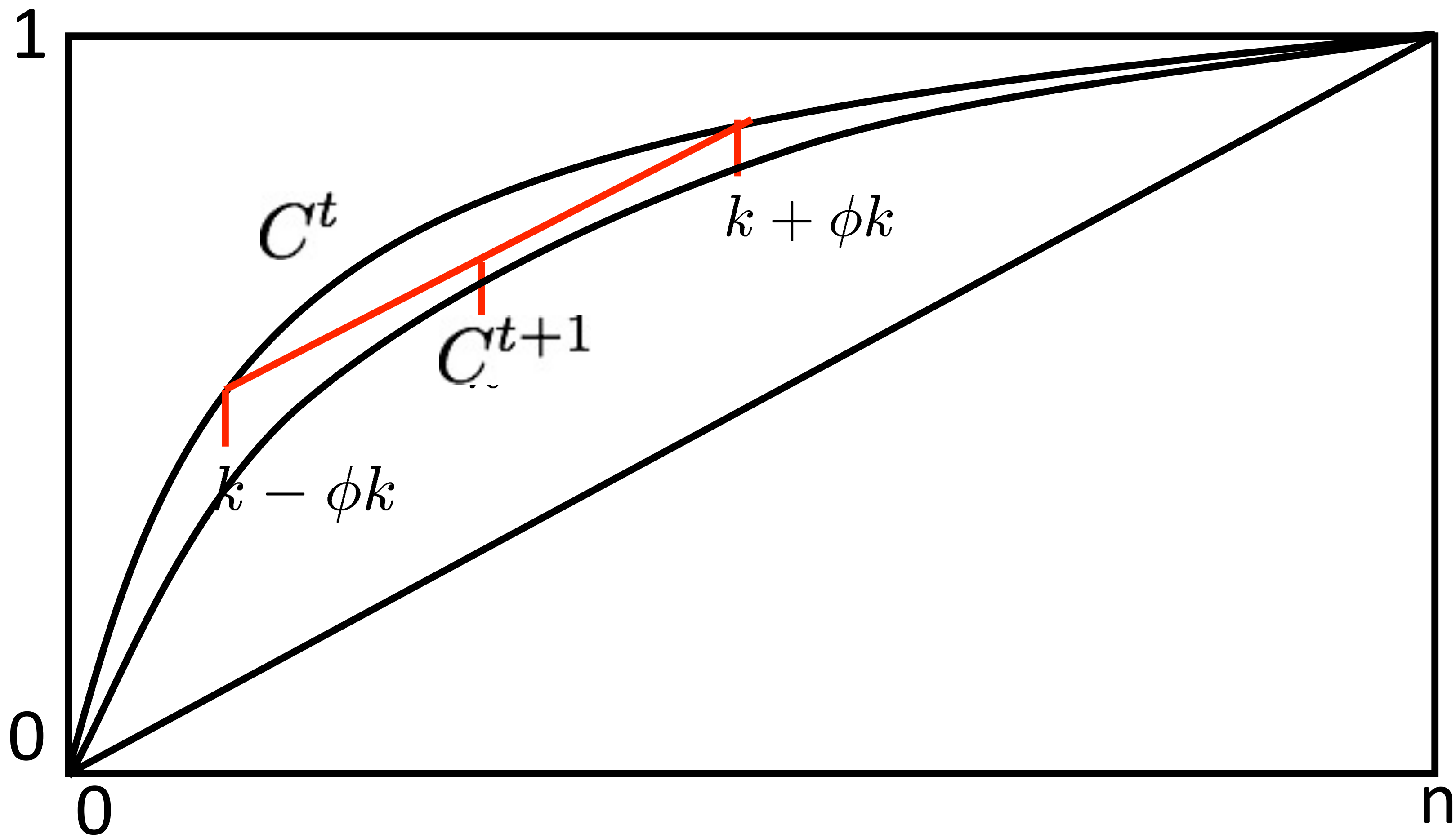
$$C^{t+1}(k) \le \frac{1}{2}\left(C^t(k-\phi(n-k)) + C^t(k+\phi(n-k))\right) \quad k \in [n/2, n]$$



Lovász-Simonovits: Conductance ϕ inequality

$$C^{t+1}(k) \leq \frac{1}{2}\left(C^t(k-\phi k) + C^t(k+\phi k)\right) \qquad k \in [0, n/2]$$

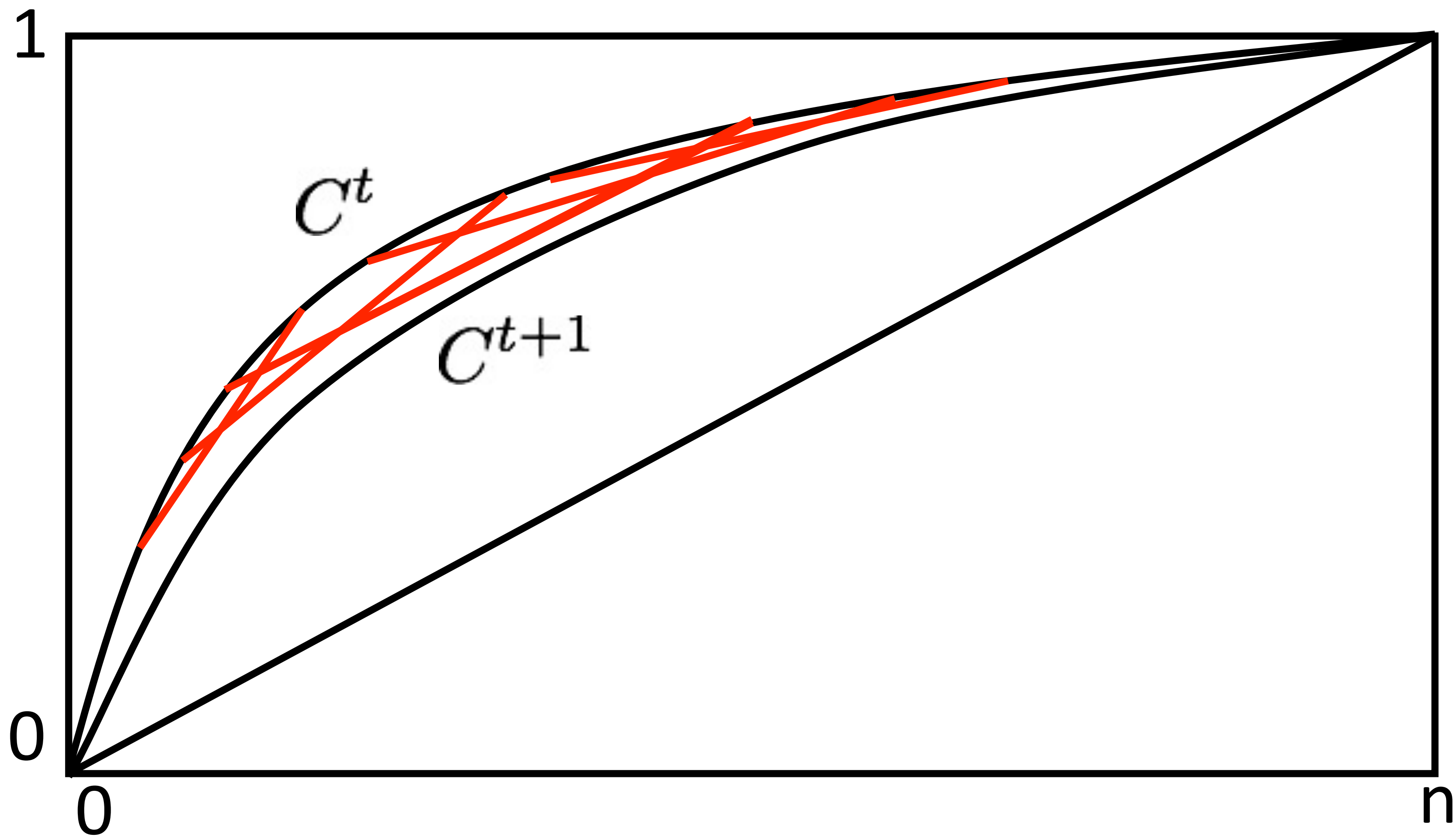
$$C^{t+1}(k) \leq \frac{1}{2}\left(C^t(k-\phi(n-k)) + C^t(k+\phi(n-k))\right) \; k \in [n/2, n]$$



Lovász-Simonovits: Conductance ϕ inequality

$$C^{t+1}(k) \le \frac{1}{2}\left(C^t(k-\phi k) + C^t(k+\phi k)\right) \qquad k \in [0, n/2]$$

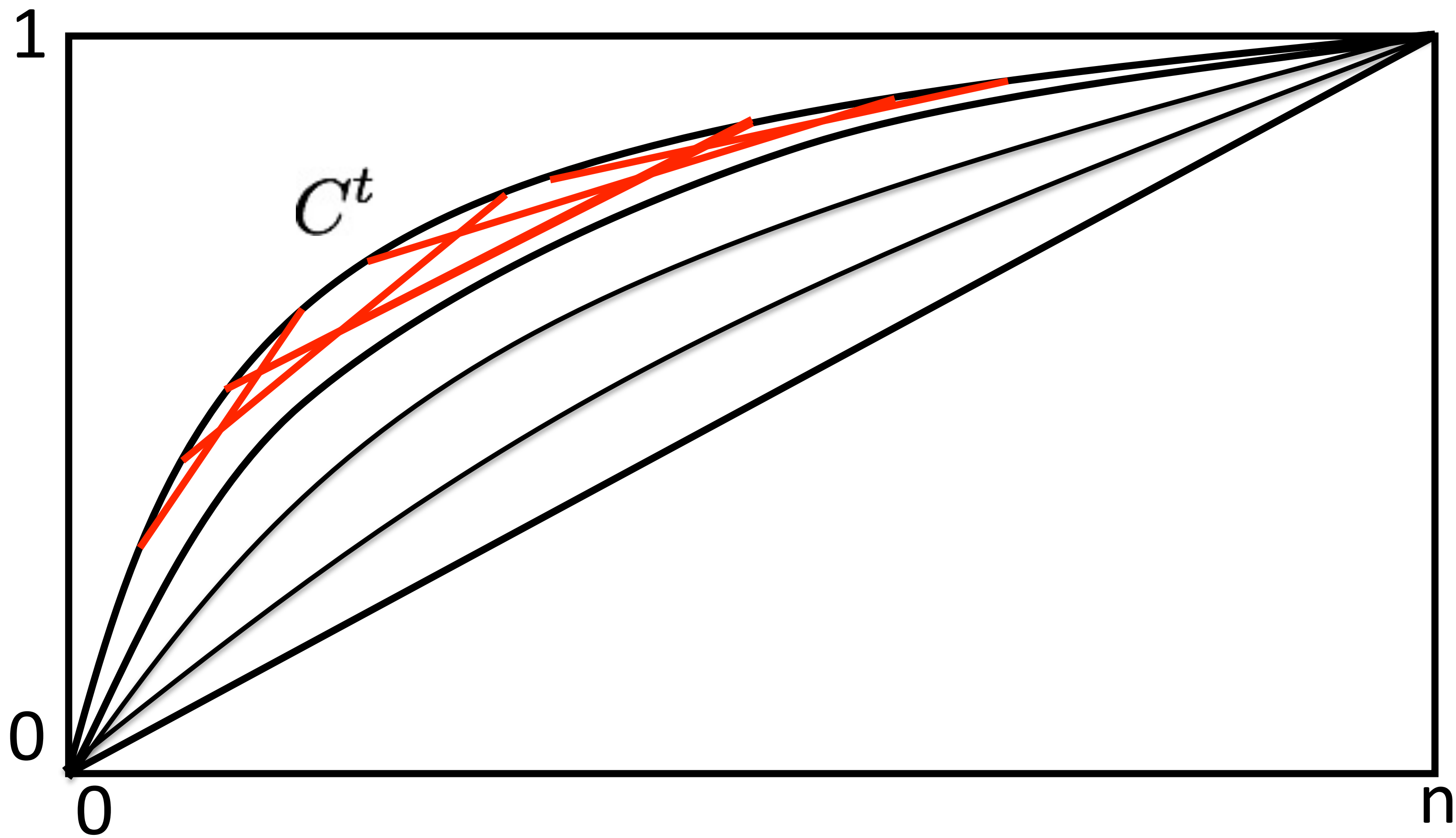
$$C^{t+1}(k) \le \frac{1}{2}\left(C^t(k-\phi(n-k)) + C^t(k+\phi(n-k))\right) \; k \in [n/2, n]$$



Lovász-Simonovits: Conductance φ inequality

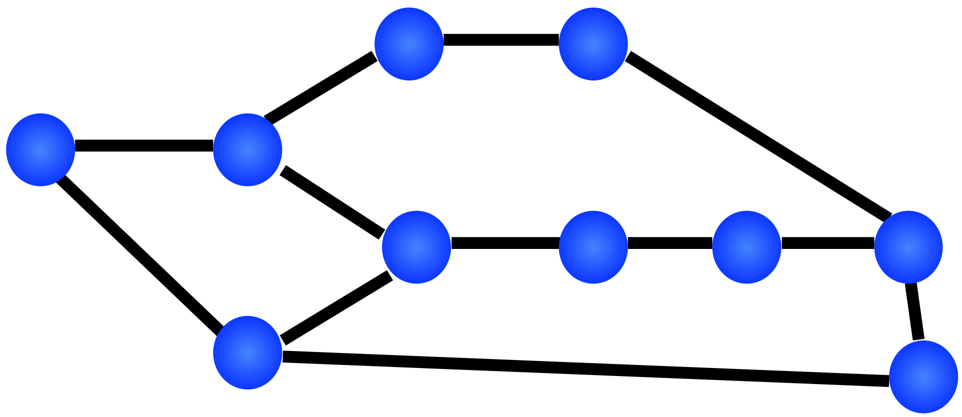
$$C^{t+1}(k) \le \frac{1}{2}\left(C^t(k-\phi k) + C^t(k+\phi k)\right) \qquad k \in [0, n/2]$$

$$C^{t+1}(k) \le \frac{1}{2}\left(C^t(k-\phi(n-k)) + C^t(k+\phi(n-k))\right) \; k \in [n/2, n]$$



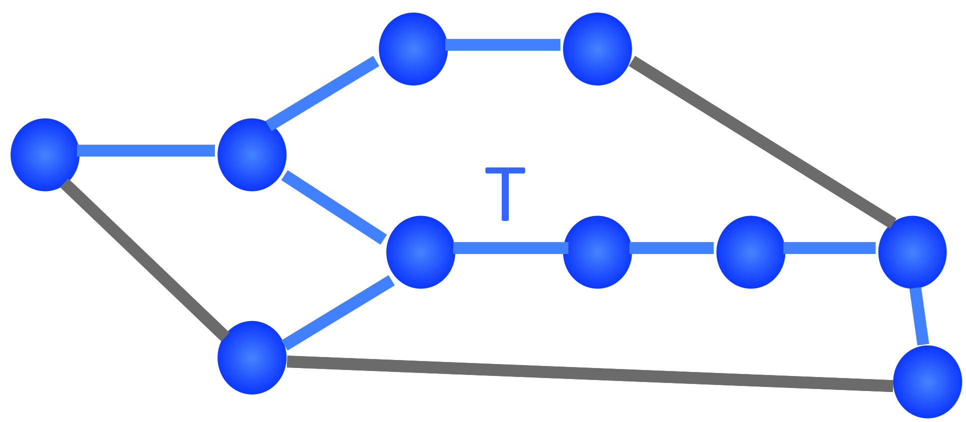
Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West '91]



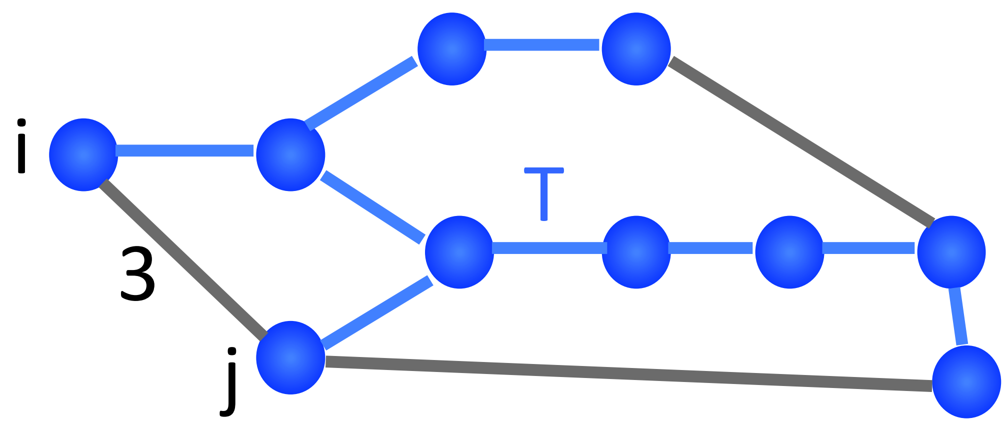
Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West '91]



Low-Stretch Spanning Trees (unweighted case)

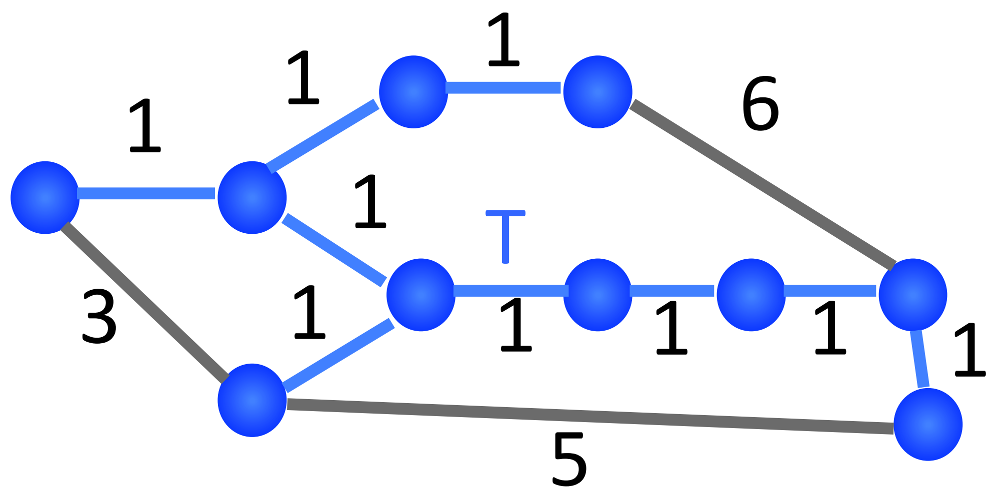
[Alon-Karp-Peleg-West '91]



$$\mathrm{stretch}_T(i,j) = \mathrm{dist}_T(i,j)$$

Low-Stretch Spanning Trees (unweighted case)

[Alon-Karp-Peleg-West '91]



$$\mathrm{stretch}_T(G) = \sum_{(i,j)\in G} \mathrm{dist}_T(i,j)$$

Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West ‘91]

$$\mathrm{stretch}_T(G) = \sum_{(i,j) \in G} \mathrm{dist}_T(i,j)/\mathrm{length}(i,j)$$

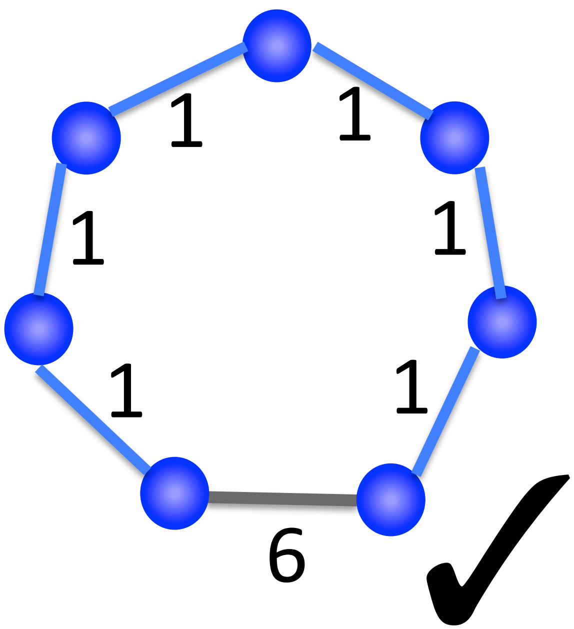
Every graph has a spanning tree of low stretch

Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West '91]

$$\mathrm{stretch}_T(G) = \sum_{(i,j)\in G} \mathrm{dist}_T(i,j)/\mathrm{length}(i,j)$$

Every graph has a spanning tree of low stretch



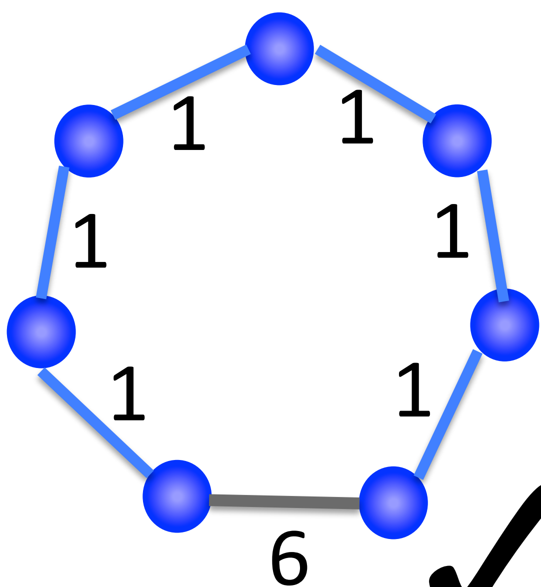
Expander:

Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West '91]

$$\mathrm{stretch}_T(G) = \sum_{(i,j) \in G} \mathrm{dist}_T(i,j)/\mathrm{length}(i,j)$$

Every graph has a spanning tree of low stretch



✓

Expander:
low diameter
use shortest path tree

✓

Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West '91]

$$\mathrm{stretch}_T(G) = \sum_{(i,j)\in G} \mathrm{dist}_T(i,j)/\mathrm{length}(i,j)$$

Every graph has a spanning tree of low stretch

$$\leq m 2^{O(\sqrt{\log n \log\log n})}$$

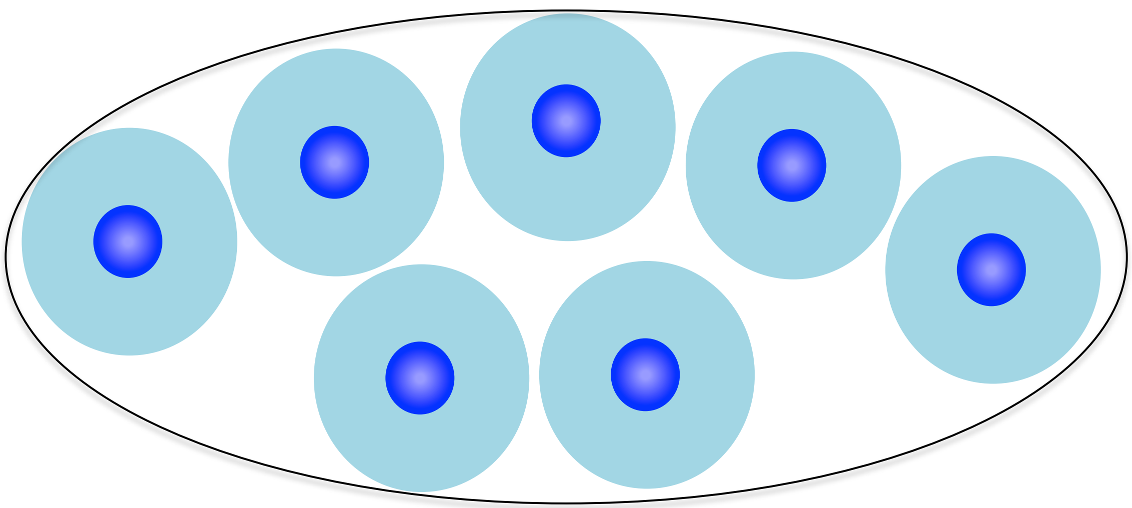
Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West '91]

Every graph has a spanning tree of low stretch

$$\leq m 2^{O(\sqrt{\log n \log \log n})}$$

Proof: by BFS/shortest paths.
Grow balls until low boundary (Awerbuch)

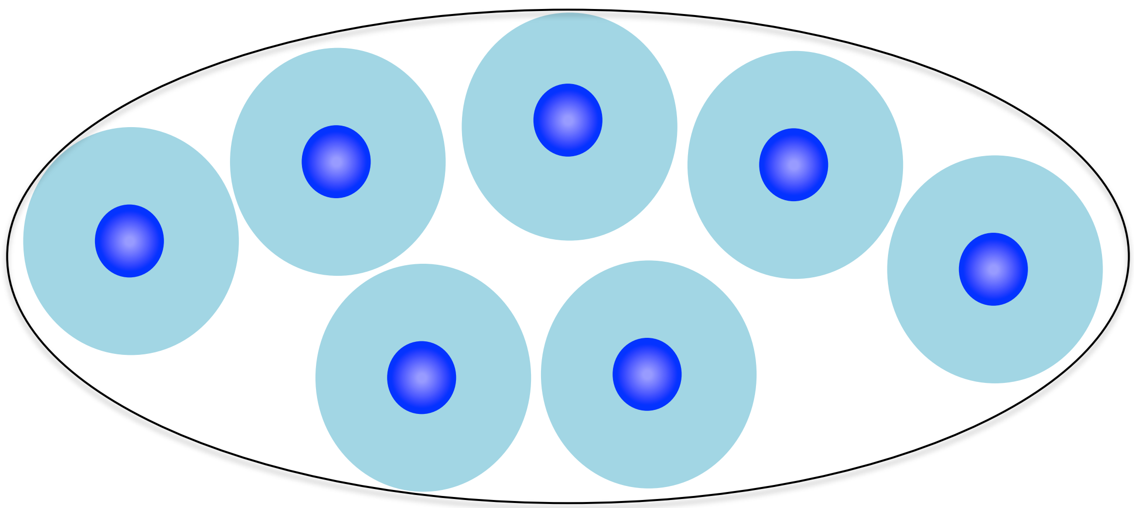


Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West '91]

Proof: by BFS/shortest paths.

Grow balls until low boundary (Awerbuch)

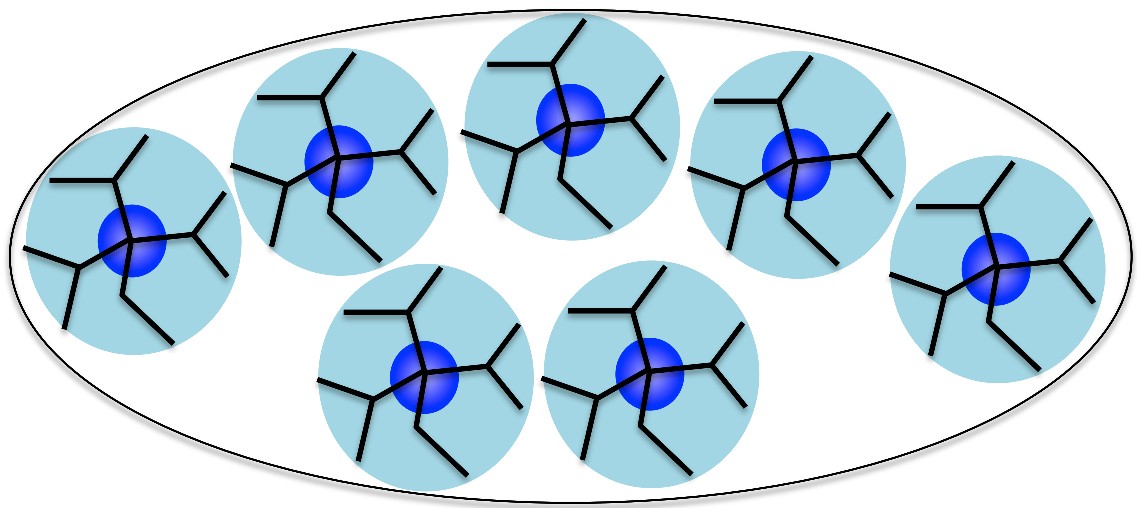


Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West '91]

Proof: by BFS/shortest paths.

Grow balls until low boundary (Awerbuch)



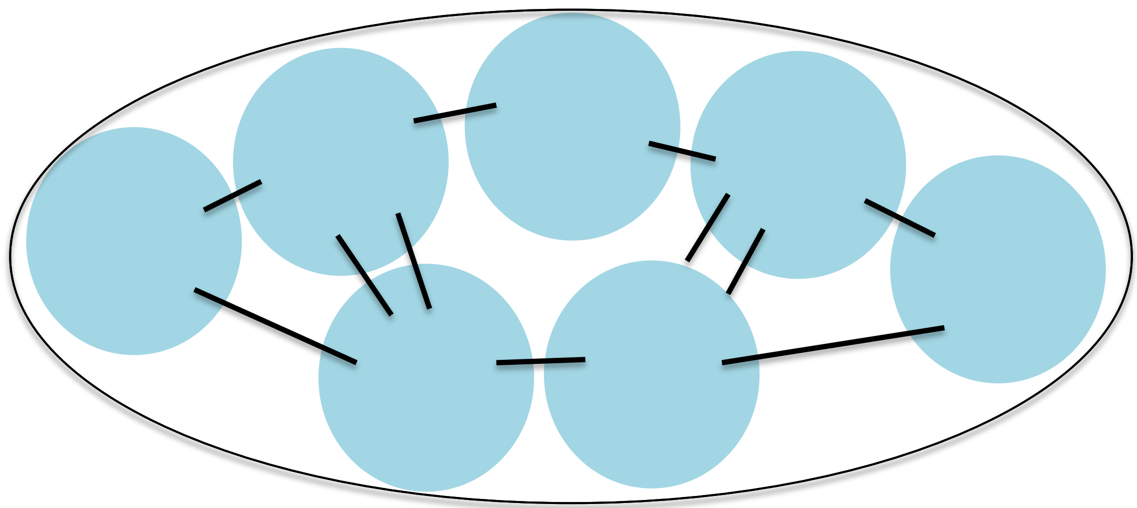
Use shortest path trees inside balls.

Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West '91]

Proof: by BFS/shortest paths.

Grow balls until low boundary (Awerbuch)



Use shortest path trees inside balls.

Contract balls.

Lower-Stretch Spanning Trees

[Elkin-Emek-S-Teng ‘04, Abraham-Bartal-Neiman ‘08]

Proof: BFS cones centered on a BFS ball.

$$\leq O(m \ \log n \ \log\log n \ (\log\log\log n)^2)$$

Lower-Stretch Spanning Trees

[Elkin-Emek-S-Teng ‘04, Abraham-Bartal-Neiman ‘08]

Proof: BFS cones centered on a BFS ball.

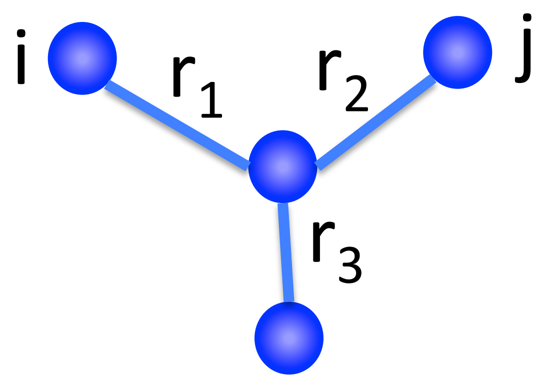
$$\leq O(m \ \log n \ \log\log n \ (\log\log\log n)^2)$$

Can we get $m \log_2 n$?

Algebraic computation of stretch [S-Woo '09]

$$\mathrm{stretch}_T(G) = \mathrm{Trace}\left[L_T^{+} L_G\right]$$

Key: in trees, resistance acts like length.



$$R_{eff}(i,j) = r_1 + r_2$$

Algebraic computation of stretch [S-Woo '09]

$$\mathrm{stretch}_T(G) = \mathrm{Trace}\left[L_T^+ L_G\right]$$

$$\begin{aligned}
\mathrm{Trace}\left[L_T^+ L_G\right] &= \mathrm{Trace}\left[L_T^+ \sum w_{i,j}(e_i - e_j)(e_i - e_j)^T\right] \\
&= \sum w_{i,j}\,\mathrm{Trace}\left[L_T^+ (e_i - e_j)(e_i - e_j)^T\right] \\
&= \sum w_{i,j}\,\mathrm{Trace}\left[(e_i - e_j)^T L_T^+ (e_i - e_j)\right] \\
&= \sum w_{i,j}(e_i - e_j)^T L_T^+ (e_i - e_j) \\
&= \sum w_{i,j} R_T^{eff}(i,j) \\
&= \sum w_{i,j}\,\mathrm{dist}_T(i,j) \\
&= \sum \mathrm{dist}_T(i,j)/\mathrm{length}(i,j)
\end{aligned}$$

Preconditioning: solve in time O(m k)

Need H $\subseteq$ G, H is a tree + m/k edges
such that $\lambda_{max}(L_H^+ L_G) \leq k^2$

Preconditioning: solve in time O(m k)

Need $H \subseteq G$, H is a tree + m/k edges such that $\lambda_{max}(L_H^+ L_G) \leq k^2$

For low-stretch T,

$$\sum_i \lambda_i(L_T^+ L_G) = \text{Trace}\left[L_T^+ L_G\right] \leq O(m \log n)$$

At most s eigenvalues larger than $O((m/s) \log n)$

Preconditioning: solve in time O(m k)

Need $H \subseteq G$, H is a tree + m/k edges
such that $\lambda_{max}(L_H^+ L_G) \leq k^2$

For low-stretch T,
At most s eigenvalues larger than $O((m/s)\log n)$

Kolla-Makarychev-Saberi-Teng '10:
Fix s eigenvalues with O(s) edges

s = m / log n   k = O(log n)

Preconditioning: solve in time O(m k)

Need H $\subseteq$ G, H is a tree + m/k edges

such that $\lambda_{max}(L_H^+ L_G) \leq k^2$

For low-stretch T,

At most s eigenvalues larger than $O((m/s)\log n)$

Kolla-Makarychev-Saberi-Teng ‘10:

Fix s eigenvalues with O(s) edges

s = m / log n k = O(log n)

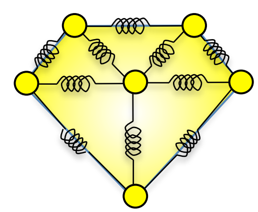
But, slow to choose edges

Preconditioning: solve in time O(m k)

Need $H \subseteq G$, H is a tree + m/k edges
such that $\lambda_{max}(L_H^+ L_G) \leq k^2$

For low-stretch T,
At most s eigenvalues larger than $O((m/s) \log n)$

Koutis-Miller-Peng '10:



$k = O(\log^2 n)$

Easy to choose edges: sample by stretch

Faster Laplacian Solvers?

KMST '09 says O(m log n) might be possible

Very powerful primitive

Like sorting

Faster Laplacian Solvers?

KMST '09 says O(m log n) might be possible

Very powerful primitive

Like sorting

Better than BFS?

Uses for better graph exploration?

Faster local clustering?

Convergence of processes on graphs?

Conclusion

We’ve produced a lot of gems