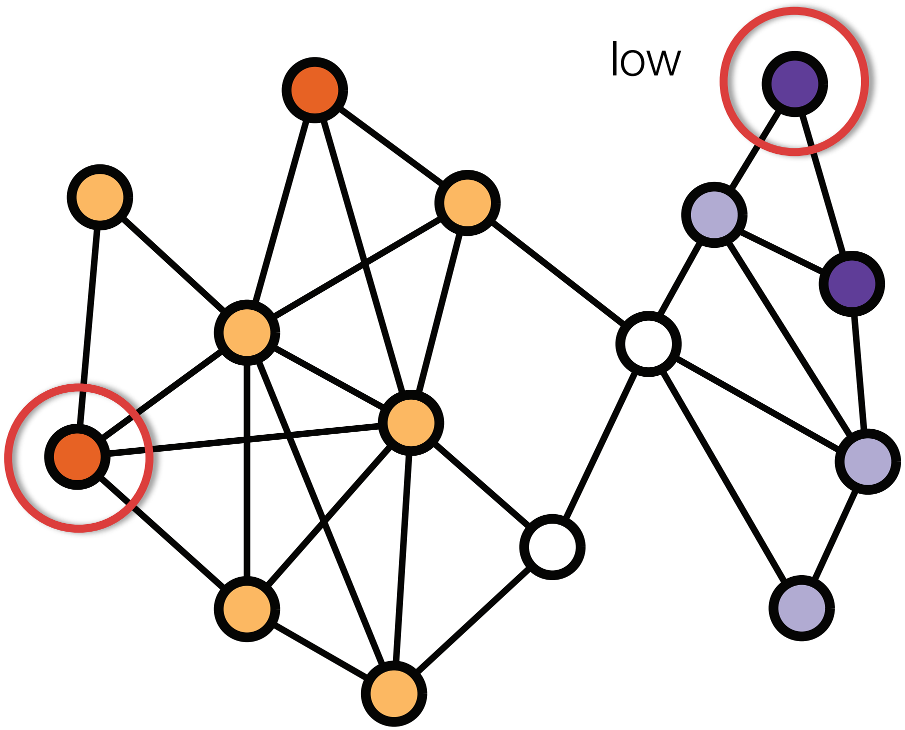
Robust spectral diffusions for data applications

Code `www.cs.purdue.edu/homes/dgleich/codes/l1pagerank`

Joint work with Michael Mahoney @ Berkeley
supported by NSF CAREER CCF-1149756

David F. Gleich
Purdue University

Graph diffusions



$$\mathbf{P} = \mathbf{A}\mathbf{D}^{-1}$$

$$\mathbf{P}\mathbf{x} = \sum_{j \to i} \frac{1}{d_j} x_j$$

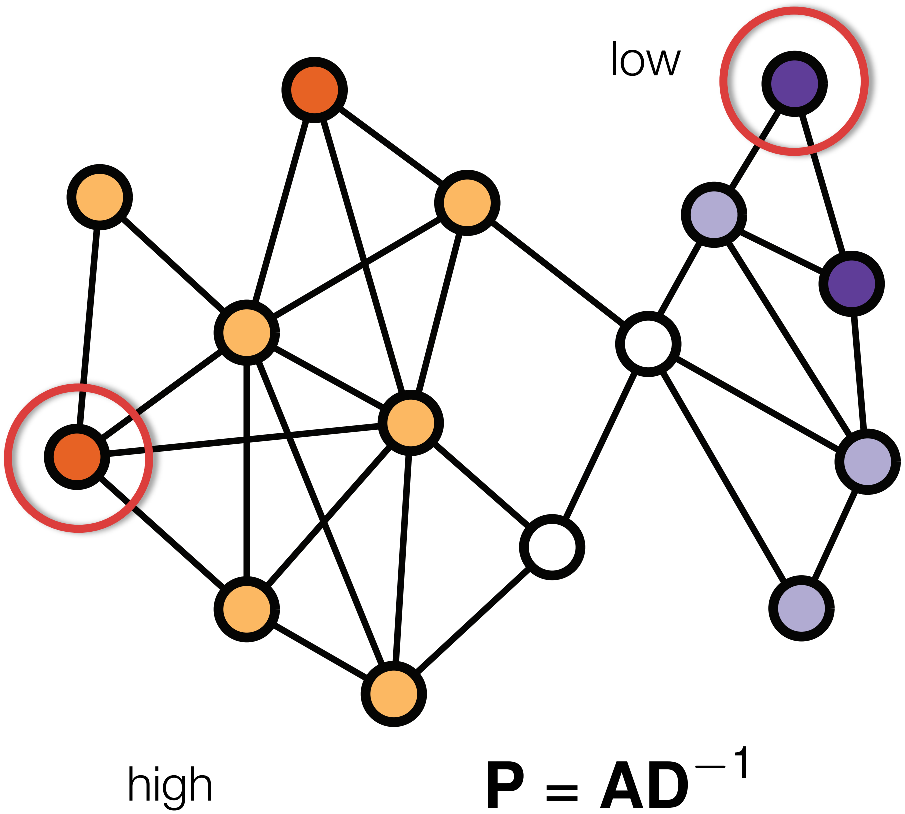
$$\mathbf{f} = \sum_{k=0}^{\infty} \alpha_k \mathbf{P}^k \mathbf{s}$$

A – adjacency matrix
D – degree matrix
P – column stochastic operator
s – the "seed" (a sparse vector)
f – the diffusion result
α_k – the path weights

Graph diffusions help:

1. Attribute prediction
2. Community detection
3. "Ranking"
4. Find small conductance sets

Practical graph diffusions



$$\mathbf{P} = \mathbf{A}\mathbf{D}^{-1}$$

$$\mathbf{P}\mathbf{x} = \sum_{j \to i} \frac{1}{d_j} x_j$$

PageRank

$$\mathbf{x} = (1-\beta)\sum_{k=0}^{\infty} \beta^k \mathbf{P}^k \mathbf{s}$$

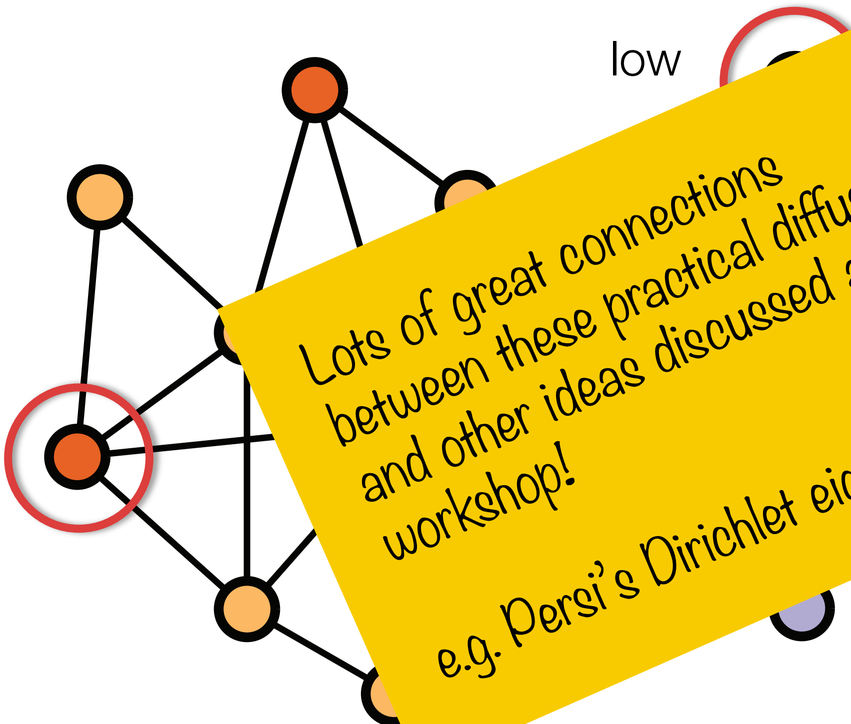
$$(\mathbf{I} - \beta\mathbf{P})\mathbf{x} = (1-\beta)\mathbf{s}$$

Heat kernel

$$\mathbf{h} = e^{-t}\sum_{k=0}^{\infty} \frac{t^k}{k!}\mathbf{P}^k \mathbf{s}$$

$$\mathbf{h} = e^{-t}\exp\{t\mathbf{P}\}\mathbf{s}$$

Practical graph diffusions



Lots of great connections between these practical diffusions and other ideas discussed at the workshop!

e.g. Persi's Dirichlet eigenvectors

$$\mathbf{P} = \mathbf{A}\mathbf{D}^{-1}$$

$$\mathbf{P}\mathbf{x} = \sum_{j \to i} \frac{1}{d_j} x_j$$

PageRank

$$\mathbf{x} = (1-\beta) \sum_{k=0}^{\infty} \beta^k \mathbf{P}^k \mathbf{s}$$

$$(\mathbf{I} - \beta \mathbf{P})\mathbf{x} = (1-\beta)\mathbf{s}$$

Heat kernel

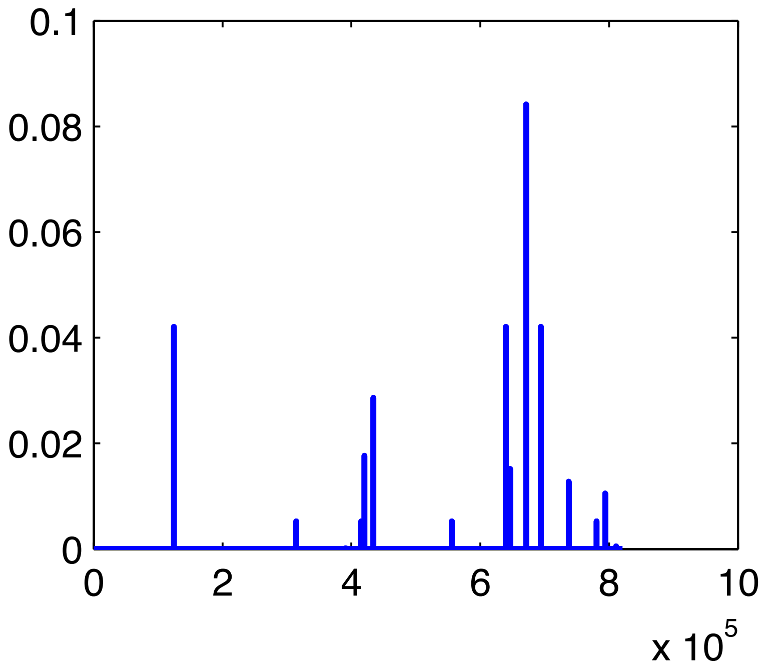
$$\mathbf{h} = e^{-t} \sum_{k=0}^{\infty} \frac{t^k}{k!} \mathbf{P}^k \mathbf{s}$$

$$\mathbf{h} = e^{-t} \exp\{t\mathbf{P}\}\mathbf{s}$$

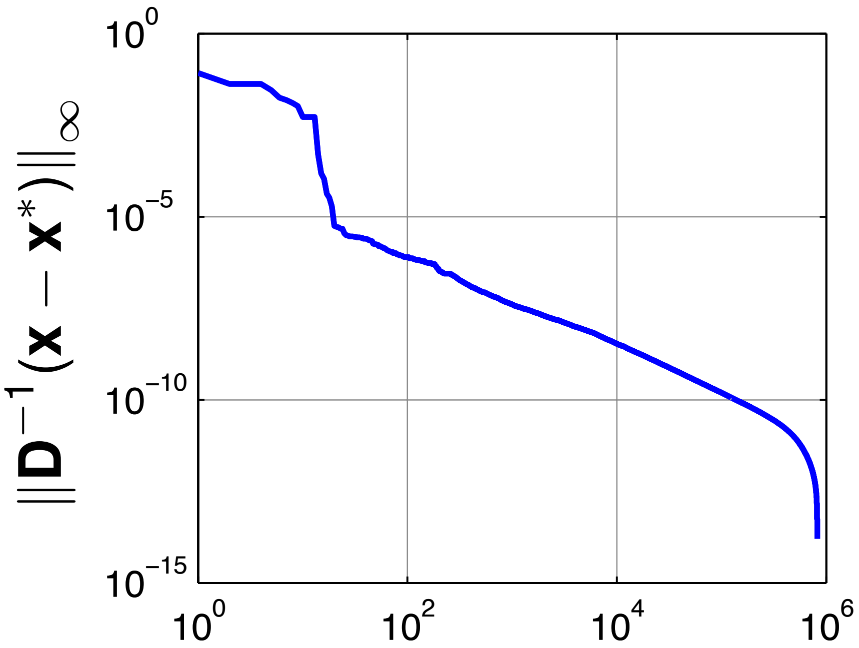
Uniformly localized solutions in flickr

$$(\mathbf{I} - \beta \mathbf{P})\mathbf{x} = (1 - \beta)\mathbf{s}$$

plot($\mathbf{x}$)



nnz($\mathbf{x}$) $\approx 800k$



Crawl of flickr from 2006 ~800k nodes, 6M edges, beta=1/2

Our mission

Understand how localization can help make diffusions robust to graph constructions and label mistakes

(and make everything faster too!)

Two types of localization

Localized vectors are not sparse, but they can be approximated by sparse vectors.

$$\mathbf{x} \approx \mathbf{x}^*$$

Other work.

Uniform (Strong)

$$\|\mathbf{x} - \mathbf{x}^*\|_1 \leq \varepsilon$$

Good global approximation using only a local region.
"Hard" to prove.
"Need" a graph property.

This talk.

Entry-wise (Weak)

$$\|\mathbf{D}^{-1}(\mathbf{x} - \mathbf{x}^*)\|_\infty \leq \varepsilon$$

Good approximation for cuts and communities.
"Easy" to prove.
"Fast" algorithms

We have three main results

1. A new interpretation for the PageRank diffusion in relationship with a mincut problem.
2. A new understanding of the scalable, localized PageRank “push” method as a regularized diffusion
3. Insights on how this regularization and graph density helps to robustify diffusions.

Undirected graphs only

Entry-wise localization

The PageRank problem & the Laplacian on undirected graphs

The PageRank random surfer

1. With probability beta, follow a random-walk step
2. With probability (1-beta), jump randomly ~ dist. **s**.

Goal find the stationary dist. **x**

1. $(\mathbf{I} - \beta \mathbf{A}\mathbf{D}^{-1})\mathbf{x} = (1 - \beta)\mathbf{s}; \quad \mathbf{x} = (1 - \beta)\sum_{k=0}^{\infty} \beta^k \mathbf{P}^k \mathbf{s}$
2. $[\alpha \mathbf{D} + \mathbf{L}]\mathbf{z} = \alpha \mathbf{s}$ where $\beta = 1/(1+\alpha)$ and $\mathbf{x} = \mathbf{D}\mathbf{z}$.

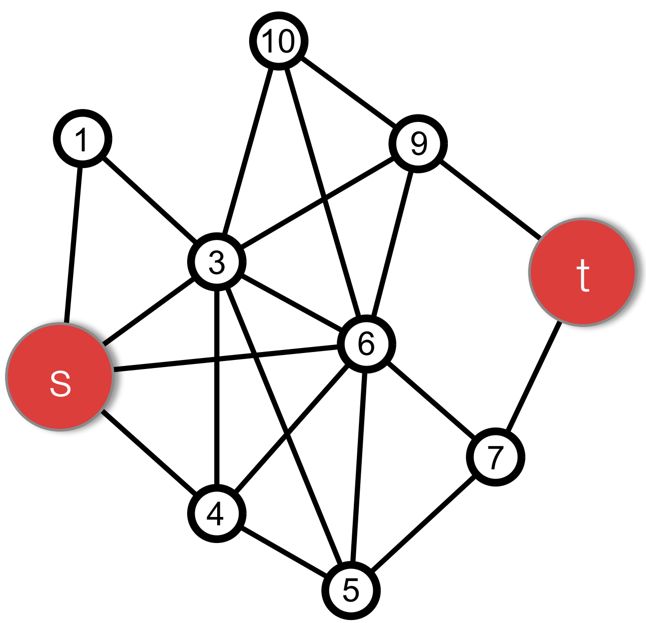
Combinatorial Laplacian **L = D - A**

The s-t min-cut problem

Unweighted incidence matrix

Diagonal capacity matrix

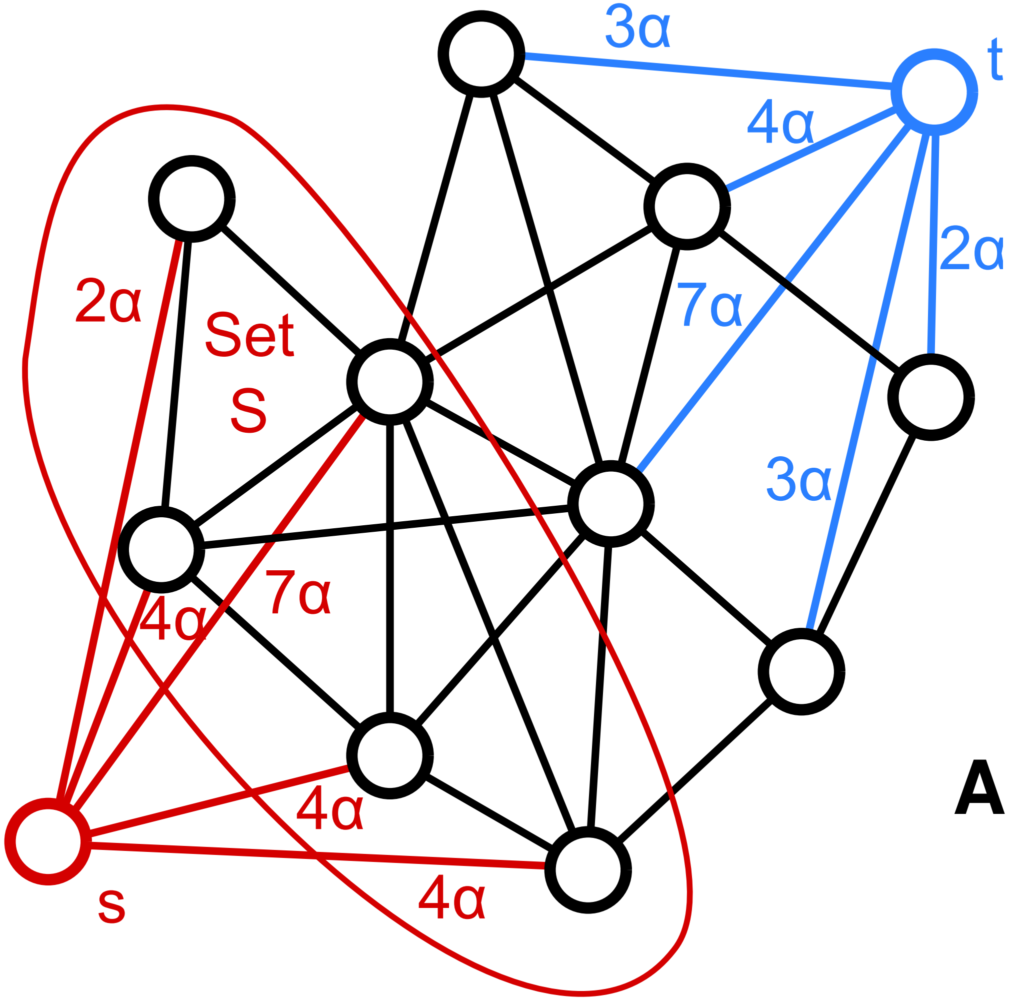
$$\begin{array}{ll} \text{minimize} & \|\mathbf{Bx}\|_{C,1} = \sum_{ij \in E} C_{i,j} |x_i - x_j| \\ \text{subject to} & x_s = 1, x_t = 0, \mathbf{x} \geq 0. \end{array}$$



In the unweighted case, solve via max-flow.

In the weighted case, solve via network simplex or industrial LP.

The localized cut graph

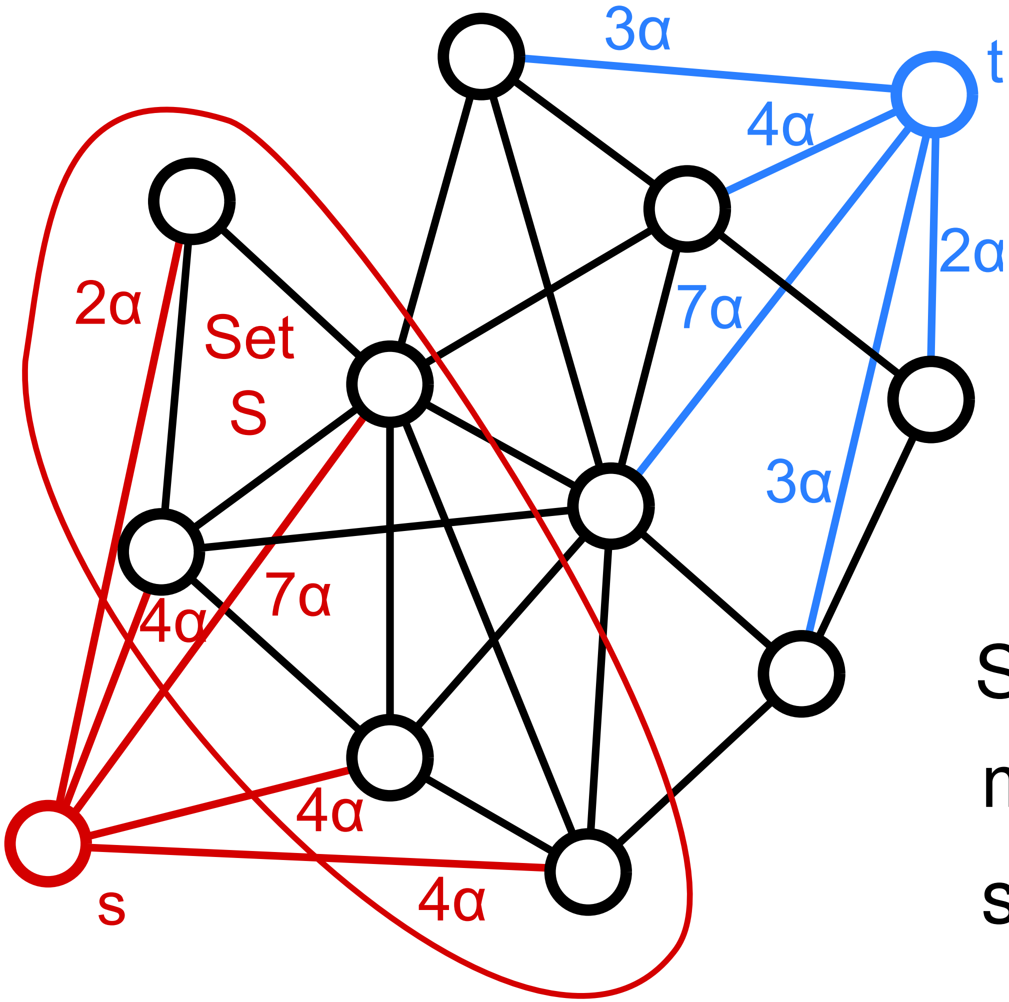


Connect s to vertices in S with weight $\alpha \cdot$ degree
Connect t to vertices in $\bar{S}$ with weight $\alpha \cdot$ degree

Related to a construction used in "FlowImprove" Andersen & Lang (2007); and Orecchia & Zhu (2014)

$$\mathbf{A}_S = \begin{bmatrix} 0 & \alpha \mathbf{d}_S^T & 0 \\ \alpha \mathbf{d}_S & \mathbf{A} & \alpha \mathbf{d}_{\bar{S}} \\ 0 & \alpha \mathbf{d}_{\bar{S}}^T & 0 \end{bmatrix}$$

The localized cut graph



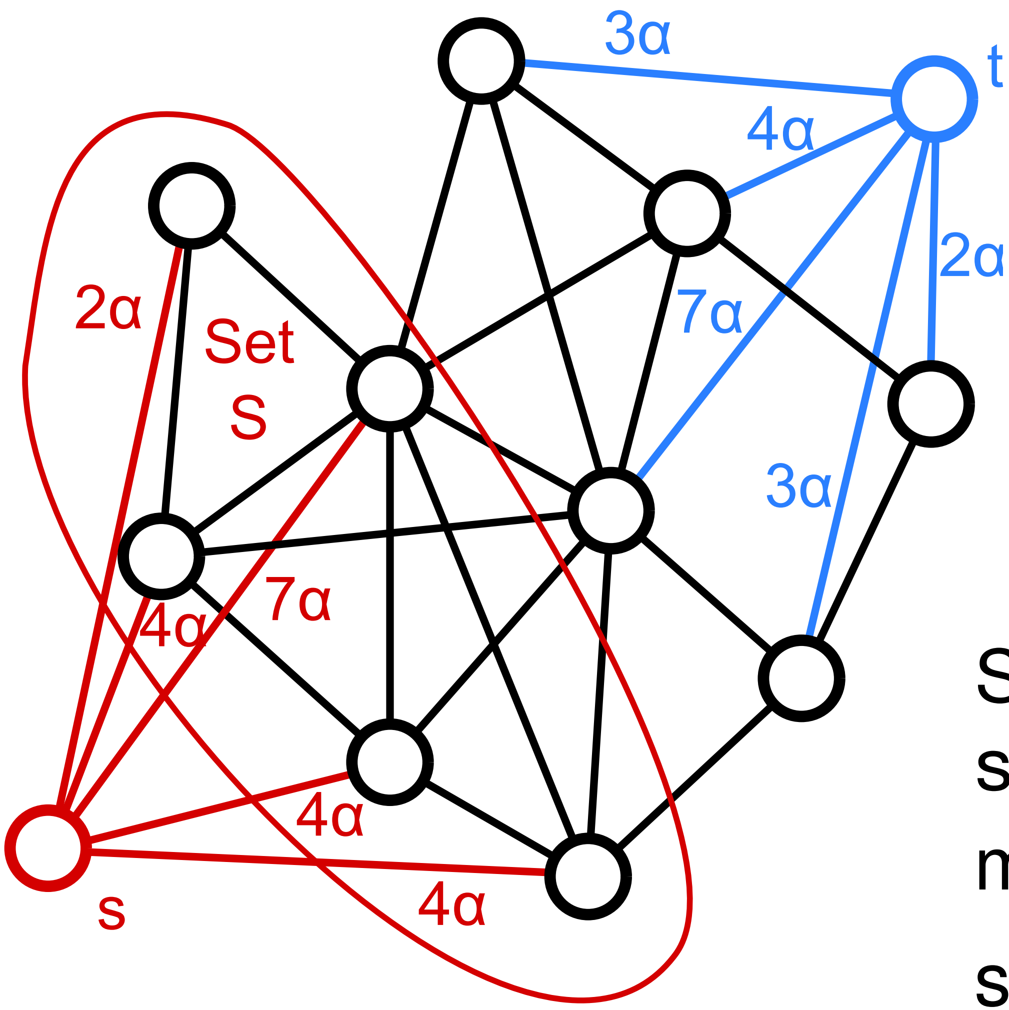
Connect *s* to vertices in *S* with weight $\alpha \cdot$ degree
Connect *t* to vertices in $\bar{S}$ with weight $\alpha \cdot$ degree

$$\mathbf{B}_S = \begin{bmatrix} \mathbf{e} & -\mathbf{I}_S & 0 \\ 0 & \mathbf{B} & 0 \\ 0 & -\mathbf{I}_{\bar{S}} & \mathbf{e} \end{bmatrix}$$

Solve the s-t min-cut

$$\begin{array}{ll} \text{minimize} & \|\mathbf{B}_S \mathbf{x}\|_{C(\alpha),1} \\ \text{subject to} & x_s = 1, x_t = 0 \\ & \mathbf{x} \geq 0. \end{array}$$

The localized cut graph



Connect s to vertices in S with weight $\alpha \cdot$ degree
Connect t to vertices in $\bar{S}$ with weight $\alpha \cdot$ degree

$$\mathbf{B}_S = \begin{bmatrix} \mathbf{e} & -\mathbf{I}_S & 0 \\ 0 & \mathbf{B} & 0 \\ 0 & -\mathbf{I}_{\bar{S}} & \mathbf{e} \end{bmatrix}$$

Solve the "electrical flow" s-t min-cut

minimize $\quad \|\mathbf{B}_S \mathbf{x}\|_{C(\alpha),2}$

subject to $\quad x_s = 1, x_t = 0$

s-t min-cut → PageRank

The PageRank vector **z** that solves

$$(\alpha \mathbf{D} + \mathbf{L})\mathbf{z} = \alpha \mathbf{s}$$

with $\mathbf{s} = \mathbf{d}_S / \text{vol}(S)$ is a renormalized solution of the electrical cut computation:

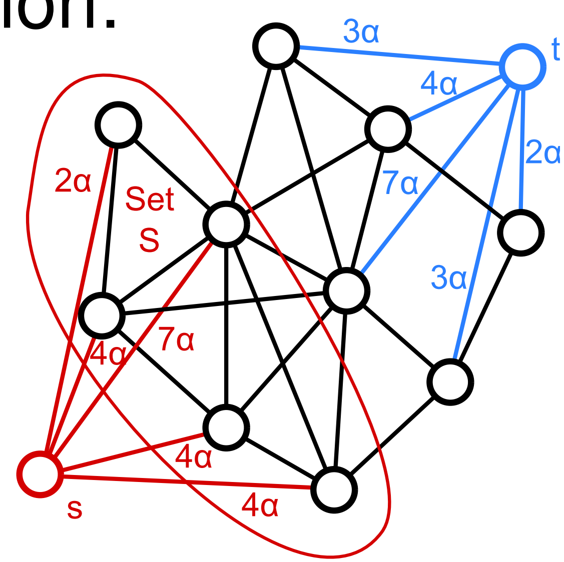
$$\begin{array}{ll} \text{minimize} & \|\mathbf{B}_S \mathbf{x}\|_{C(\alpha),2} \\ \text{subject to} & x_s = 1, x_t = 0. \end{array}$$

Specifically, if **x** is the solution, then

$$\mathbf{x} = \begin{bmatrix} 1 \\ \text{vol}(S)\mathbf{z} \\ 0 \end{bmatrix}$$

Proof

Square and expand the objective into a Laplacian, then apply constraints.

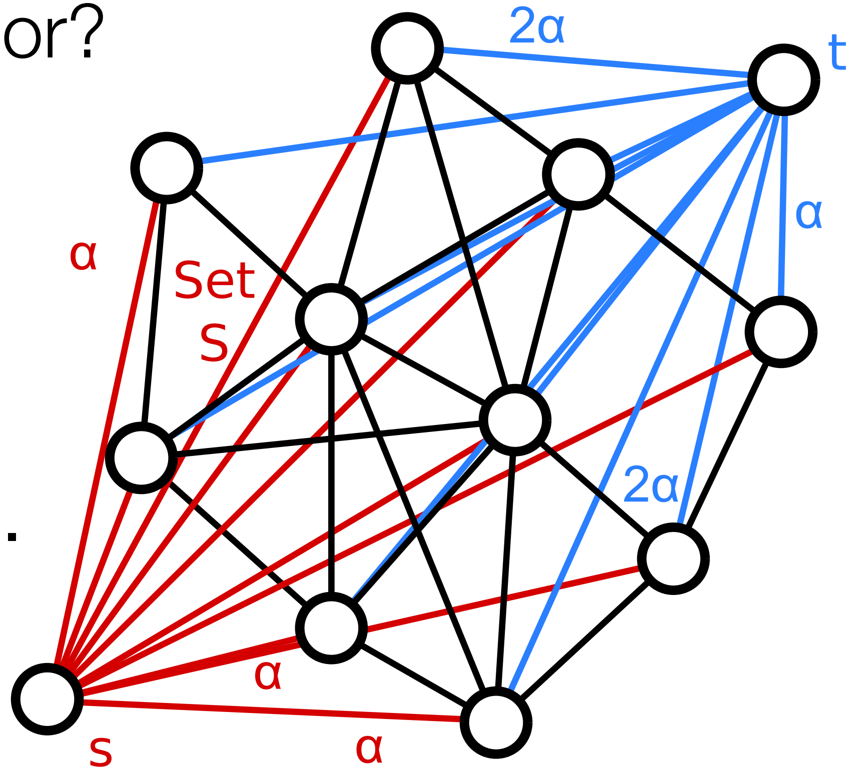


PageRank → s-t min-cut

That equivalence works if **s** is degree-weighted.

What if **s** is the uniform vector?

$$\mathbf{A}(\mathbf{s}) = \begin{bmatrix} 0 & \alpha \mathbf{s}^T & 0 \\ \alpha \mathbf{s} & \mathbf{A} & \alpha(\mathbf{d} - \mathbf{s}) \\ 0 & \alpha(\mathbf{d} - \mathbf{s})^T & 0 \end{bmatrix}.$$

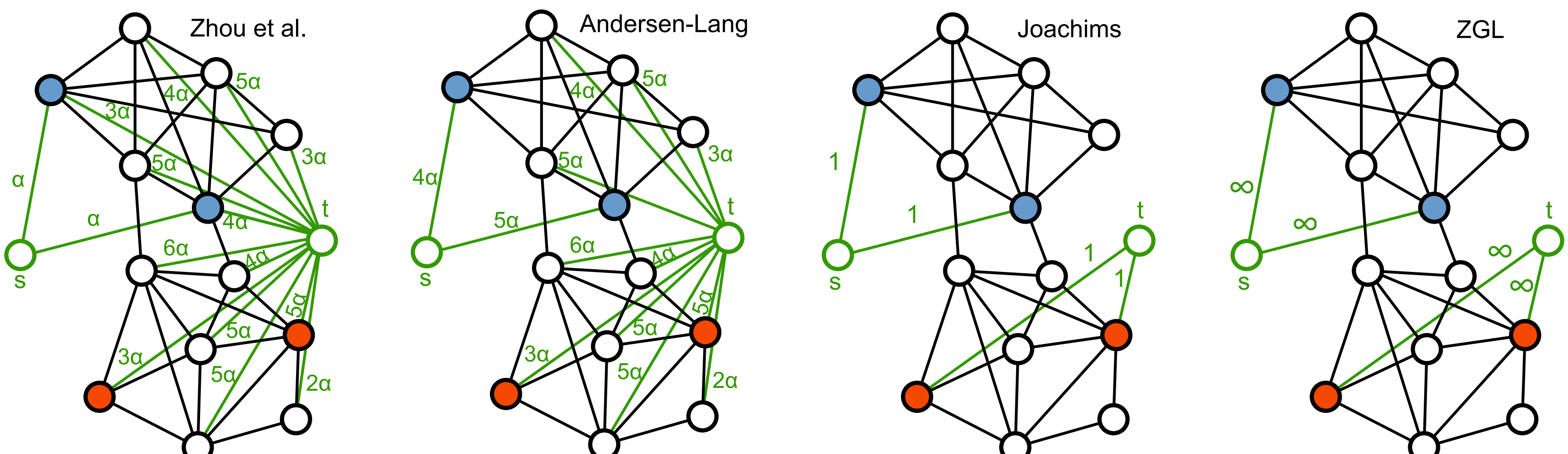


Insight 1

PageRank implicitly approximates the solution of these s-t mincut problems

Insight I'

This holds for a variety of diffusion methods for semi-supervised learning.



(a) Zhou [4] (b) Andersen-Lang [13] (c) Joachims [5] (d) ZGL [3]

Seeds have weight 1.

Seeds have weight d_i.

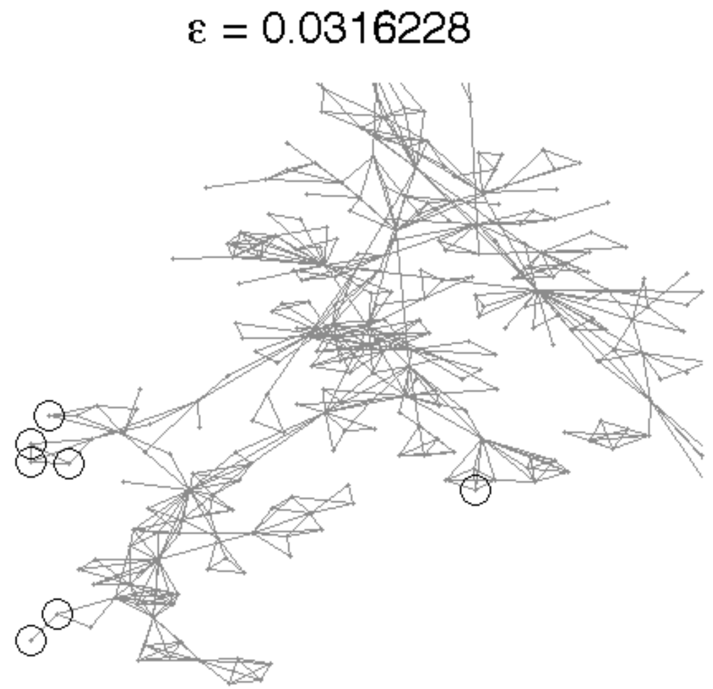
Labeled nodes have edges to source/sink. ZGL pins them

The Push Algorithm for PageRank

Proposed (in closest form) in Andersen, Chung, Lang (also by McSherry, Jeh & Widom, Berkhin) for fast approx. *PageRank*

Derived to show improved runtime for balanced solvers

1. Used for empirical studies of "communities"
2. Local Cheeger inequality.
3. Used for "fast Page-Rank approximation"
4. Works on massive graphs O(1 second) for 4 billion edge graph on a laptop.
5. It yields weakly *localized* PageRank approximations!



Newman's netscience
379 vertices, 1828 nnz

Produce an ε-accurate entrywise localized PageRank vector in work $\frac{1}{\varepsilon(1-\beta)}$

Gauss-Seidel and Gauss-Southwell

Methods to solve $\boldsymbol{A}\,\mathbf{x} = \mathbf{b}$

Update $\quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \rho_j \mathbf{e}_j \quad$ such that $\quad [\boldsymbol{A}\mathbf{x}^{(k+1)}]_j = [\mathbf{b}]_j$

In words "Relax" or "free" the *j*th coordinate of your solution vector in order to satisfy the *j*th equation of your linear system.

Gauss-Seidel repeatedly cycle through j = 1 to n

Gauss-Southwell use the value of j that has the highest magnitude residual

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$$

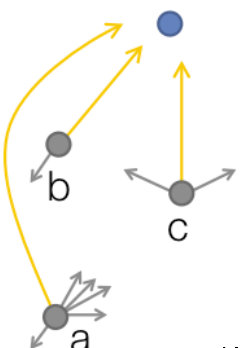
PageRank Pull and Push for Gauss-Southwell/Seidel

Different version of PageRank linear system

w/ access to in-links & degs.

PageRankPull

j = blue node   Solve for $x_j^{(k+1)}$

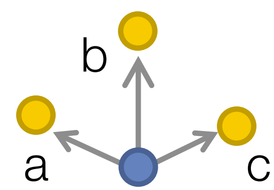


$$x_j^{(k+1)} - \alpha x_a^{(k)}/6 - \alpha x_b^{(k)}/2 - \alpha x_c^{(k)}/3 = f_j$$

$$x_j^{(k+1)} - \alpha \sum_{i \to j} x_i^{(k)}/deg_i = f_j$$

w/ access to out-l

PageRankPush



j = blue node

Let

$$\mathbf{r}^{(k)} = \mathbf{f} + \alpha \mathbf{A}^T \mathbf{D}^{-1} \mathbf{x}^{(k)} - \mathbf{x}^{(k)}$$

then $x_j^{(k+1)} = x_j^{(k)} + r_j$

Update $\mathbf{r}^{(k+1)}$ $r_j^{(k+1)} = 0$

$$r_a^{(k+1)} = r_a^{(k)} + \alpha r_j^{(k)}/3$$

$$r_b^{(k+1)} = r_b^{(k)} + \alpha r_j^{(k)}/3$$

$$r_c^{(k+1)} = r_c^{(k)} + \alpha r_j^{(k)}/3$$

Almost "the push" method

The Push Method

ε, ρ

1. $\mathbf{x}^{(1)} = 0, \mathbf{r}^{(1)} = (1-\beta)\mathbf{e}_i, k = 1$
2. *while* any $r_j > \varepsilon d_j$ (d_j *is the degree of node j*)
3. $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + (r_j - \varepsilon d_j \rho)\mathbf{e}_j$
4. $\mathbf{r}_i^{(k+1)} = \begin{cases} \varepsilon d_j \rho & i = j \\ r_i^{(k)} + \beta(r_j - \varepsilon d_j \rho)/d_j & i \sim j \\ r_i^{(k)} & \text{otherwise} \end{cases}$
5. $k \leftarrow k + 1$

Only push "some" of the residual – If we want tolerance "eps" then push to tolerance "eps" and no further

The push method revisited

Let $\mathbf{x}$ be the output from the push method with $0 < \beta < 1$, $\mathbf{v} = \mathbf{d}_S / \text{vol}(S)$, $\rho = 1$, and $\tau > 0$.

Set $\alpha = \frac{1-\beta}{\beta}$, $\kappa = \tau \text{vol}(S)/\beta$, and let $\mathbf{z}_G$ solve:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathbf{B}_S\mathbf{z}\|_{C(\alpha),2}^2 + \kappa\|\mathbf{D}\mathbf{z}\|_1 \\ \text{subject to} & z_s = 1, z_t = 0, \mathbf{z} \geq 0 \end{array},$$

Need for normalization

Regularization for sparsity

where $\mathbf{z} = \begin{bmatrix} 1 \\ \mathbf{z}_G \\ 0 \end{bmatrix}$.

Then $\mathbf{x} = \mathbf{D}\mathbf{z}_G/\text{vol}(S)$.

Proof Write out KKT conditions
Show that the push method solves them. Slackness was "tricky"

Insight 2

The PageRank push method implicitly solves a 1-norm regularized 2-norm cut approximation.

Insight 2'

We get 3-digits of accuracy on P and
16-digits of accuracy on P'.

Insight 2”

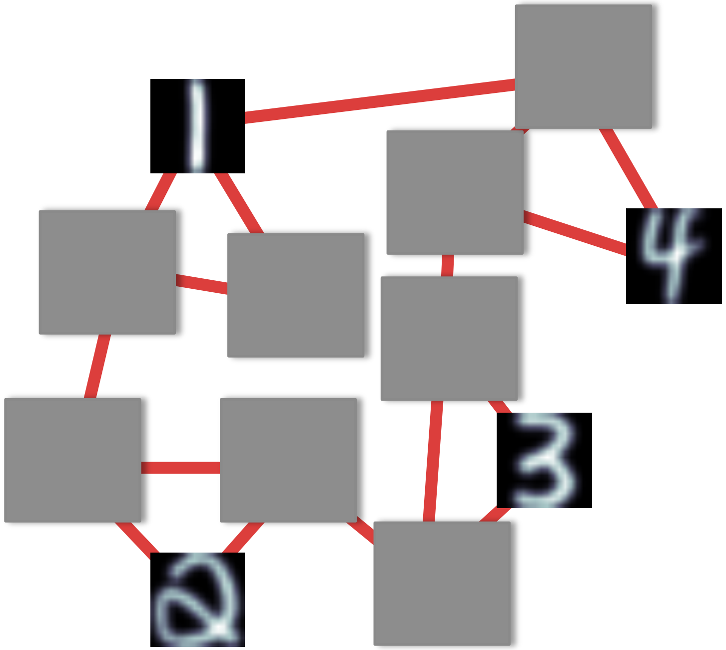
These regularized diffusions (via push) should be more robust in data applications (and faster)!

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathbf{B}_S\mathbf{z}\|^2_{C(\alpha),2} + \kappa\|\mathbf{Dz}\|_1 \\ \text{subject to} & z_s = 1, z_t = 0, \mathbf{z} \geq 0 \end{array},$$

Semi-supervised & diffusion-based learning

Gleich & Mahoney, In prep.

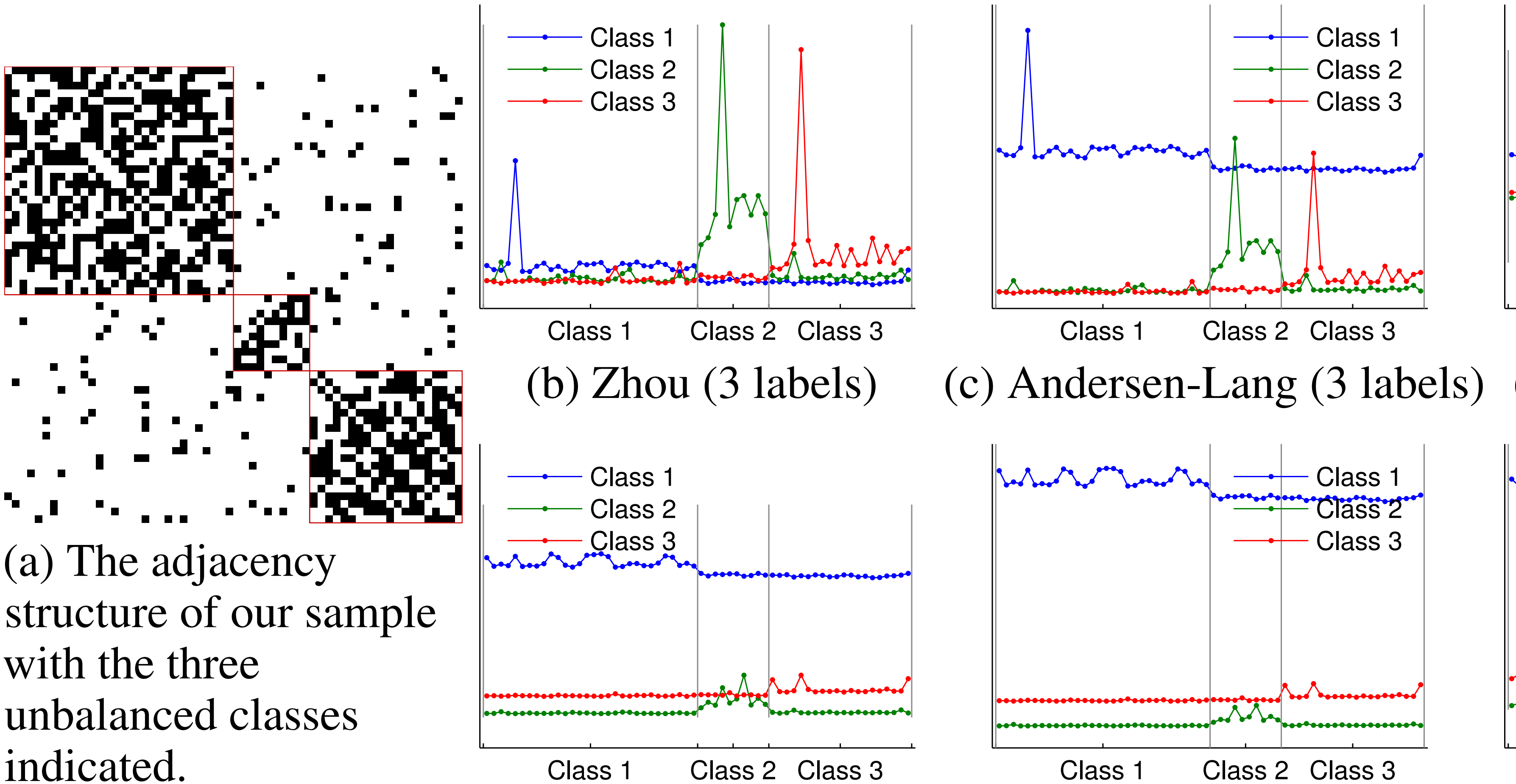
Given a graph, and a few labeled nodes, predict the labels on the rest of the graph.



Algorithm

1. Run a diffusion for each label (possibly with neg. info from other classes)
2. Assign new labels based on the value of each diffusion

Vanilla SSL algorithms have a problem



(a) The adjacency structure of our sample with the three unbalanced classes indicated.

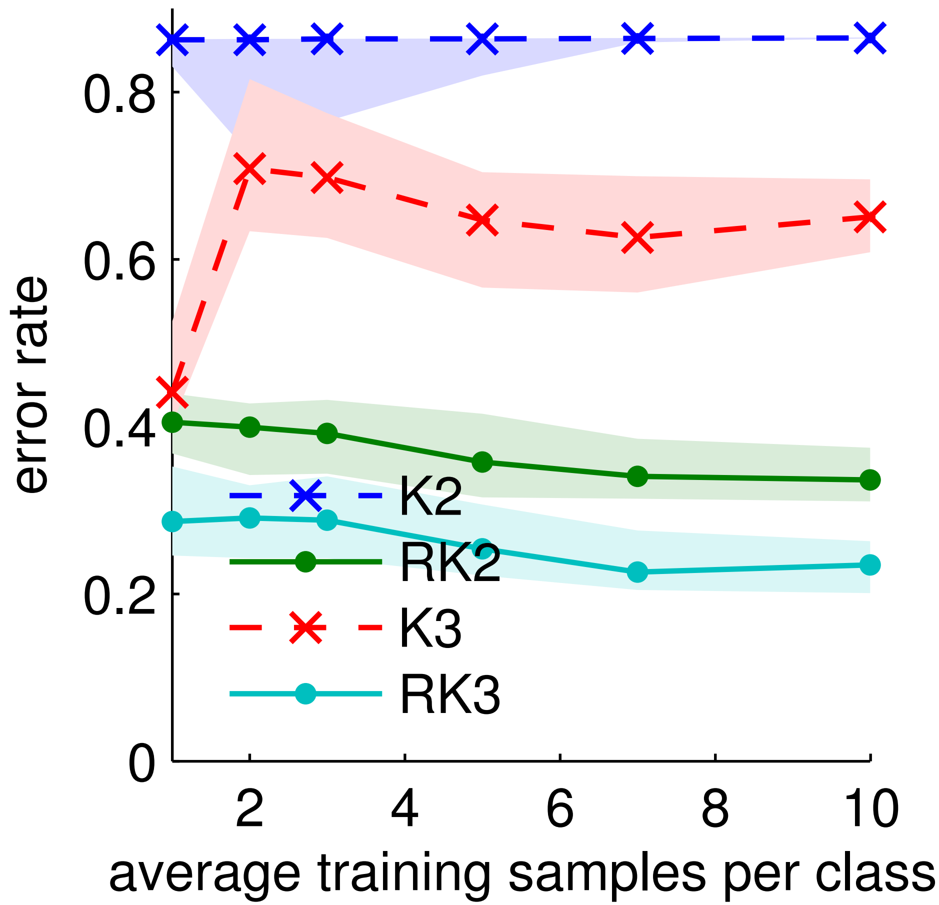
(b) Zhou (3 labels)

(c) Andersen-Lang (3 labels)

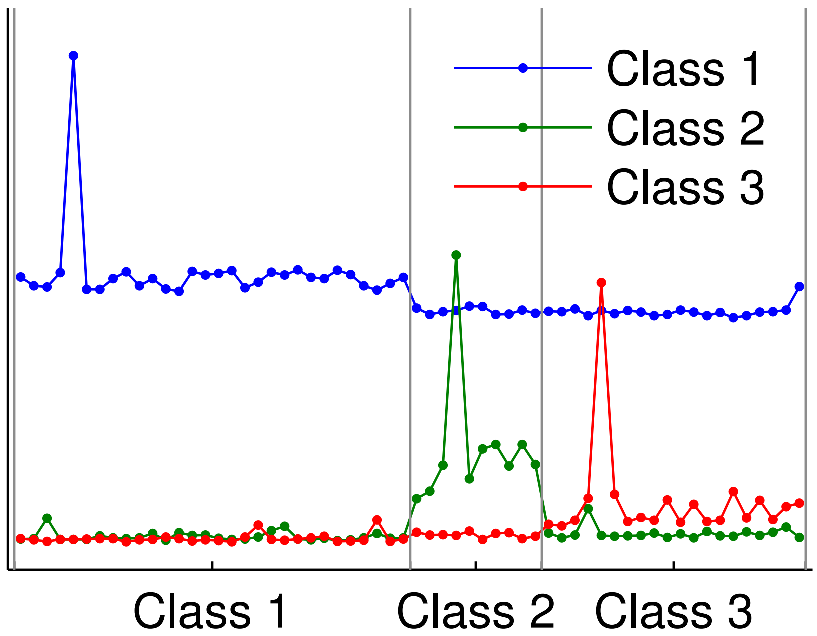
(e) Zhou (15 labels)

(f) Andersen-Lang (15 labels)

This problem is worse on real data



Unifying theory and practice



(c) Andersen-Lang (3 labels)

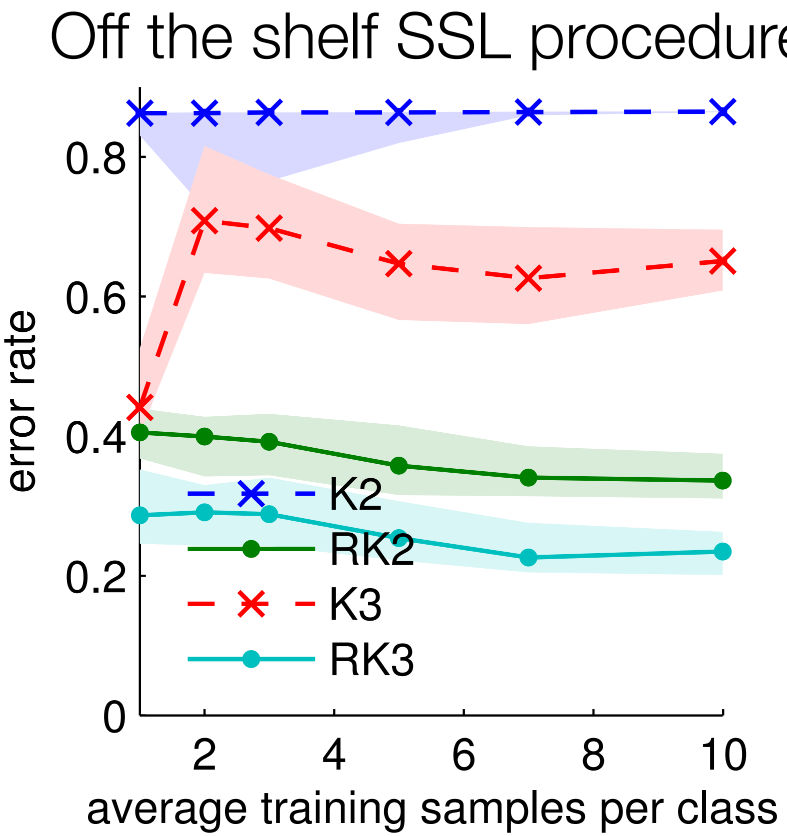
Insight 1'

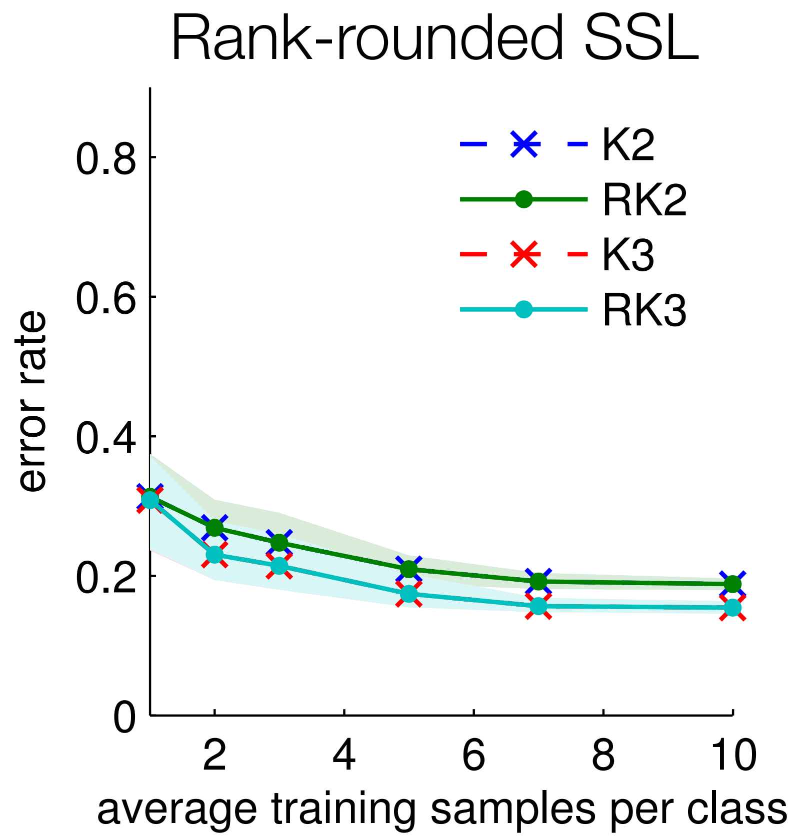
Diffusions are all approximations to cuts.

In spectral theory

We "sweep" over cuts from approximate eigenvectors!

Without these insights, we'd draw the wrong conclusion.

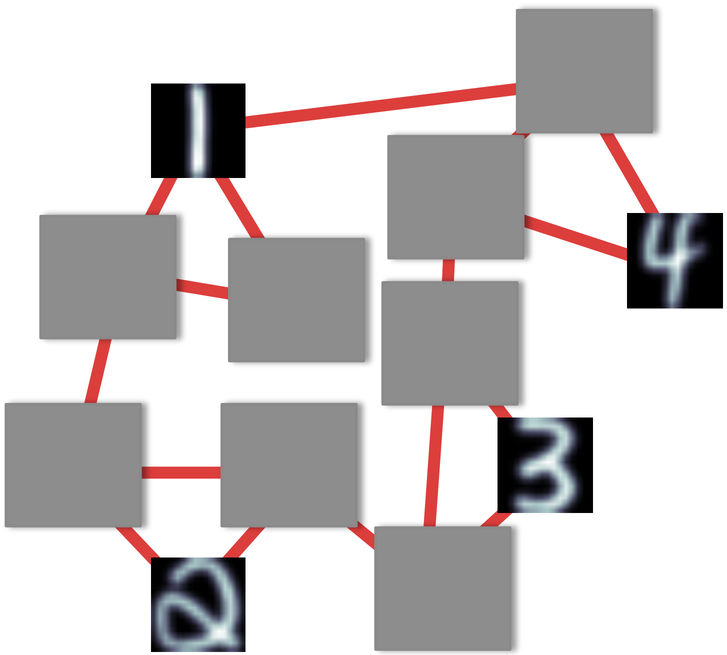




One more step …

Gleich & Mahoney,
In prep.

Given a graph, and a few labeled nodes,
predict the labels on the rest of the graph.



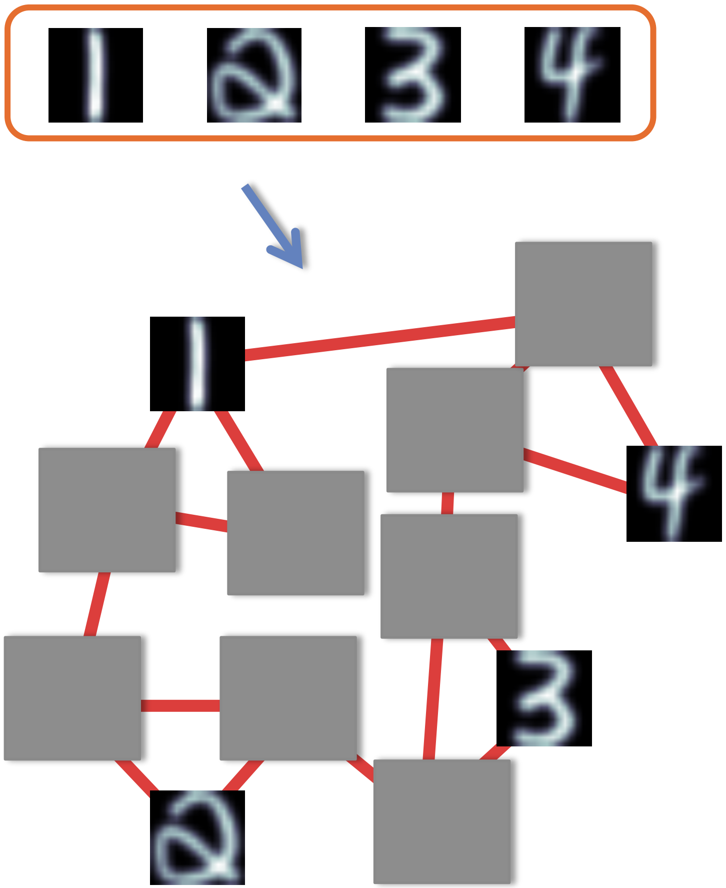
Algorithm

1. Run a diffusion for each label (possibly with neg. info from other classes)
2. Assign new labels based on the value of each diffusion

Gleich & Mahoney,
In prep.

One more step …

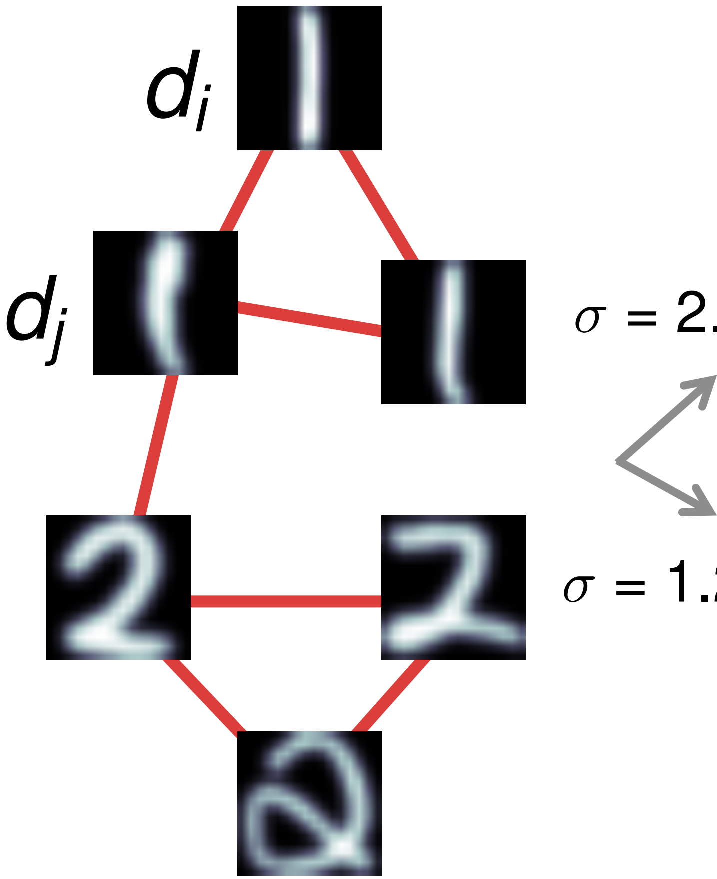
Given ~~a graph~~ data, and a few labeled ~~nodes~~ points,
predict the labels on the rest of the ~~graph~~. data



Algorithm

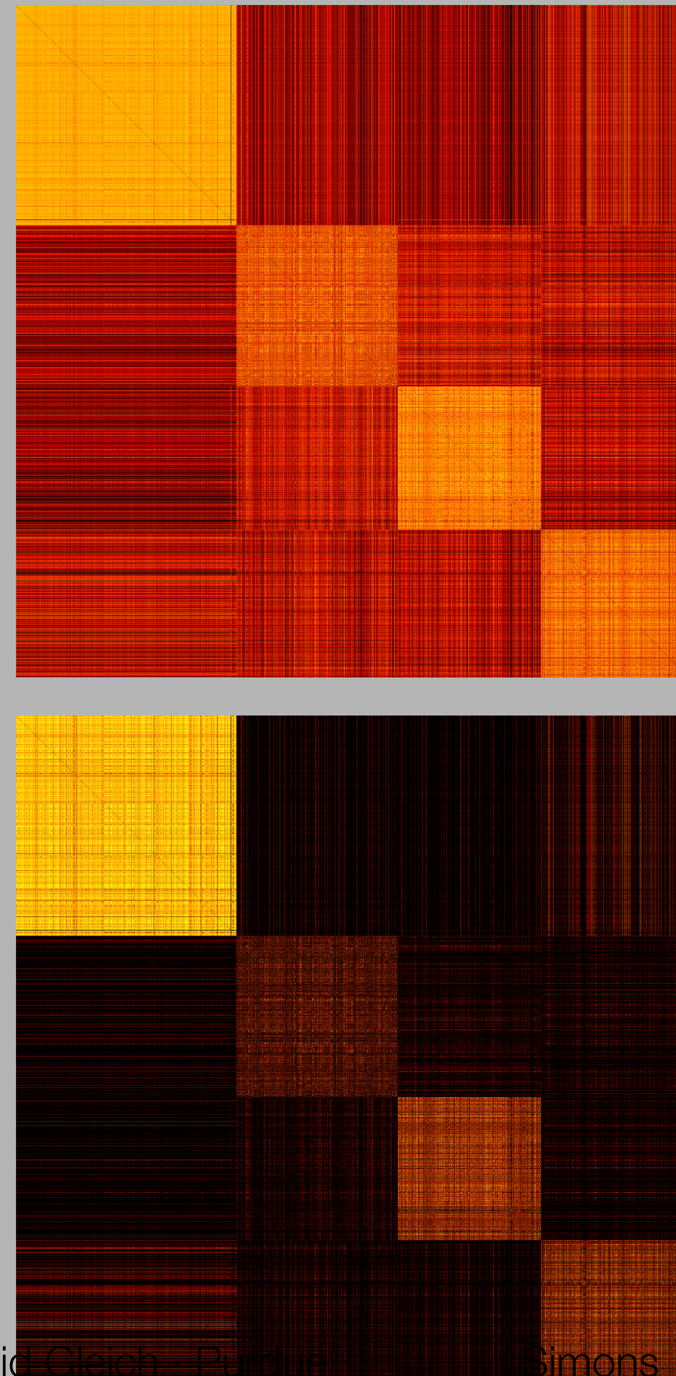
0. Create a graph from the data
1. Run a diffusion for each label (possibly with neg. info from other classes)
2. Assign new labels based on the value of each diffusion

Semi-supervised Learning on Graphs



$\sigma = 2.5$

$\sigma = 1.25$



$$A_{i,j} = \exp\left(-\frac{\|d_i - d_j\|_2^2}{2\sigma^2}\right)$$

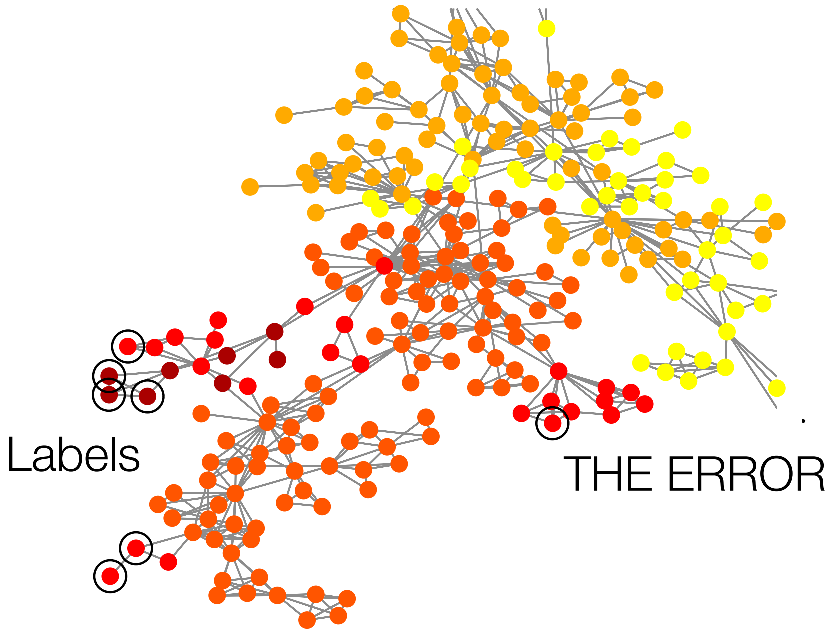
Zhou et al. NIPS (2003)

Does regularization help with sparse or dense graphs?

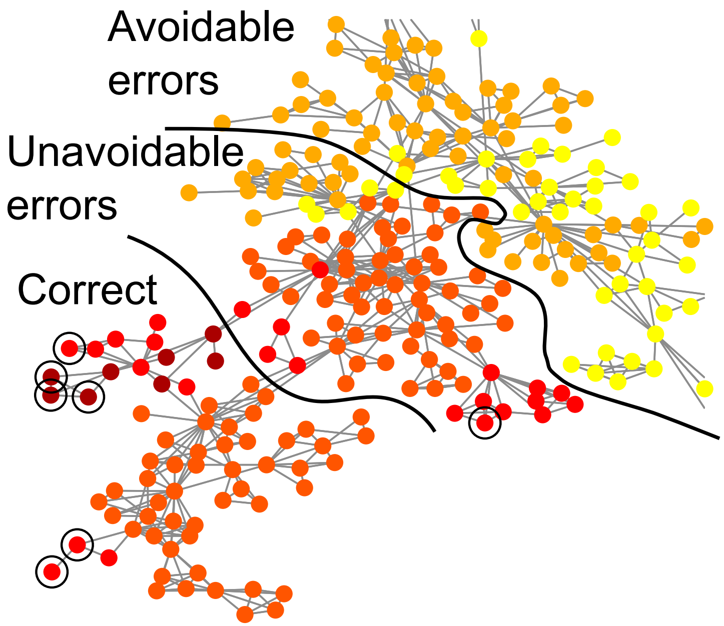
We introduce a few labeling mistakes

	σ	Method	Average training labels per class			
			2	5	7	10
Sparse	0.8	RK2	0.34%	0.22%	0.25%	-0.02%
	0.8	RK3	0.5%	0.39%	1.1%	1.1%
	1.25	RK2	0.34%	0.41%	0.24%	0.22%
	1.25	RK3	0.4%	0.39%	0.36%	0.42%
Dense	2.5	RK2	0%	0%	0%	0%
	2.5	RK3	0%	0%	0%	0%

(c) Median improvement to error rate with regularization for digit prediction; various σ and 20% label mistakes

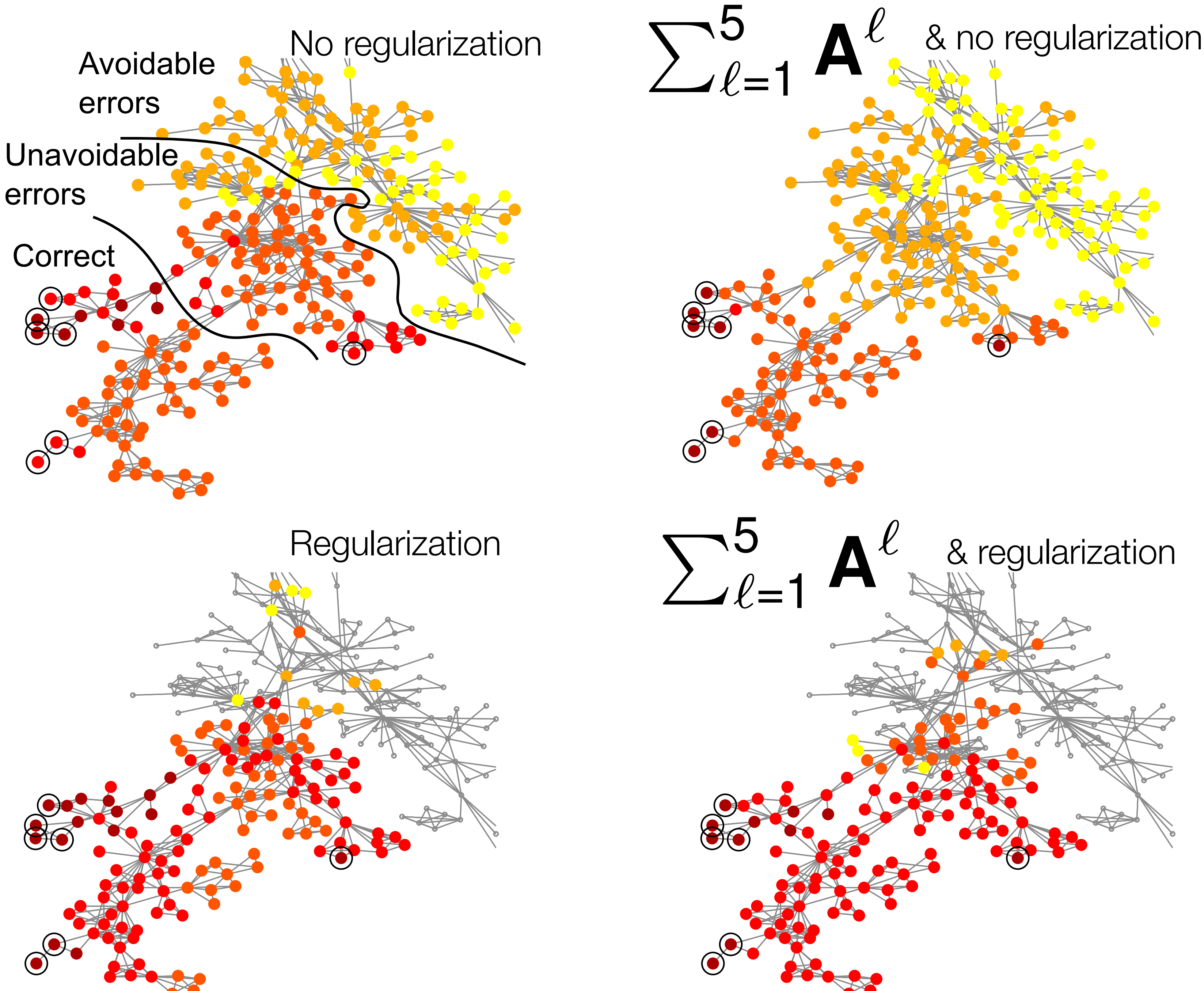


How do sparsity, density, and regularization of a diffusion play into the results in a controlled setting?



How do we take a graph and make it more dense?

$$\sum_{\ell=1}^{k} \mathbf{A}^{\ell}$$



No regularization
Avoidable errors
Unavoidable errors
Correct
$\sum_{\ell=1}^{5} \mathbf{A}^{\ell}$ & no regularization
Regularization
$\sum_{\ell=1}^{5} \mathbf{A}^{\ell}$ & regularization

Summary of robust diffusions

1. Use rank-based rounding
2. Use denser graphs if there are errors (if you can afford it).

    We are trying to get some theory to quantify this effect

    This makes computation expensive!

References

Gleich and Mahoney – Algorithmic Anti-differentiation, ICML 2014
Gleich and Mahoney – Regularized diffusions, In prep

`www.cs.purdue.edu/homes/dgleich/codes/l1pagerank`