Geometric Tools for Identifying Structure in Large Social and Information Networks

Michael W. Mahoney

Stanford University

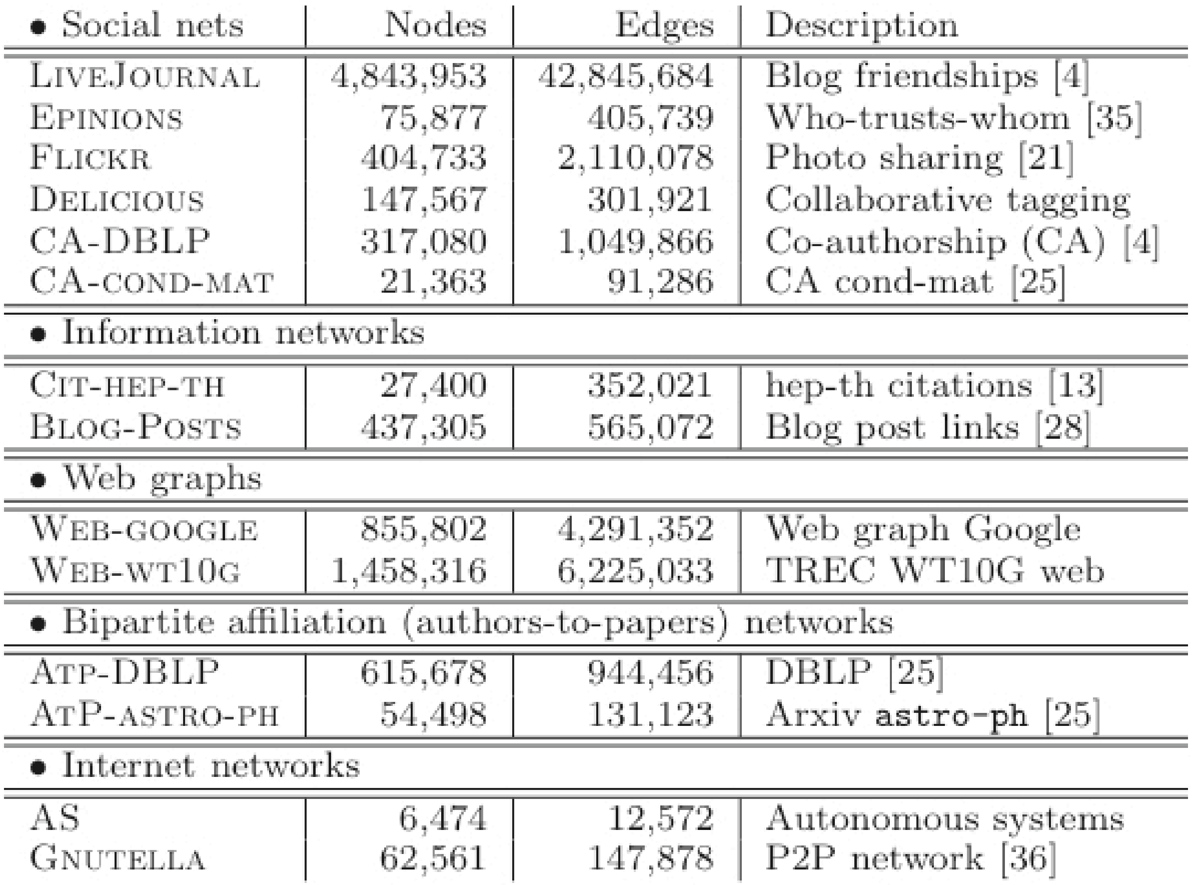
(ICML 2010 and KDD 2010 Tutorial)

(For more info, see:
http:// cs.stanford.edu/people/mmahoney/
or Google on "Michael Mahoney")

Lots of “networked data” out there!

- Technological and communication networks
  - AS, power-grid, road networks
- Biological and genetic networks
  - food-web, protein networks
- Social and information networks
  - collaboration networks, friendships; co-citation, blog cross-postings, advertiser-bidded phrase graphs ...
- Financial and economic networks
  - encoding purchase information, financial transactions, etc.
- Language networks
  - semantic networks ...
- Data-derived “similarity networks”
  - recently popular in, e.g., “manifold” learning
- ...

Large Social and Information Networks

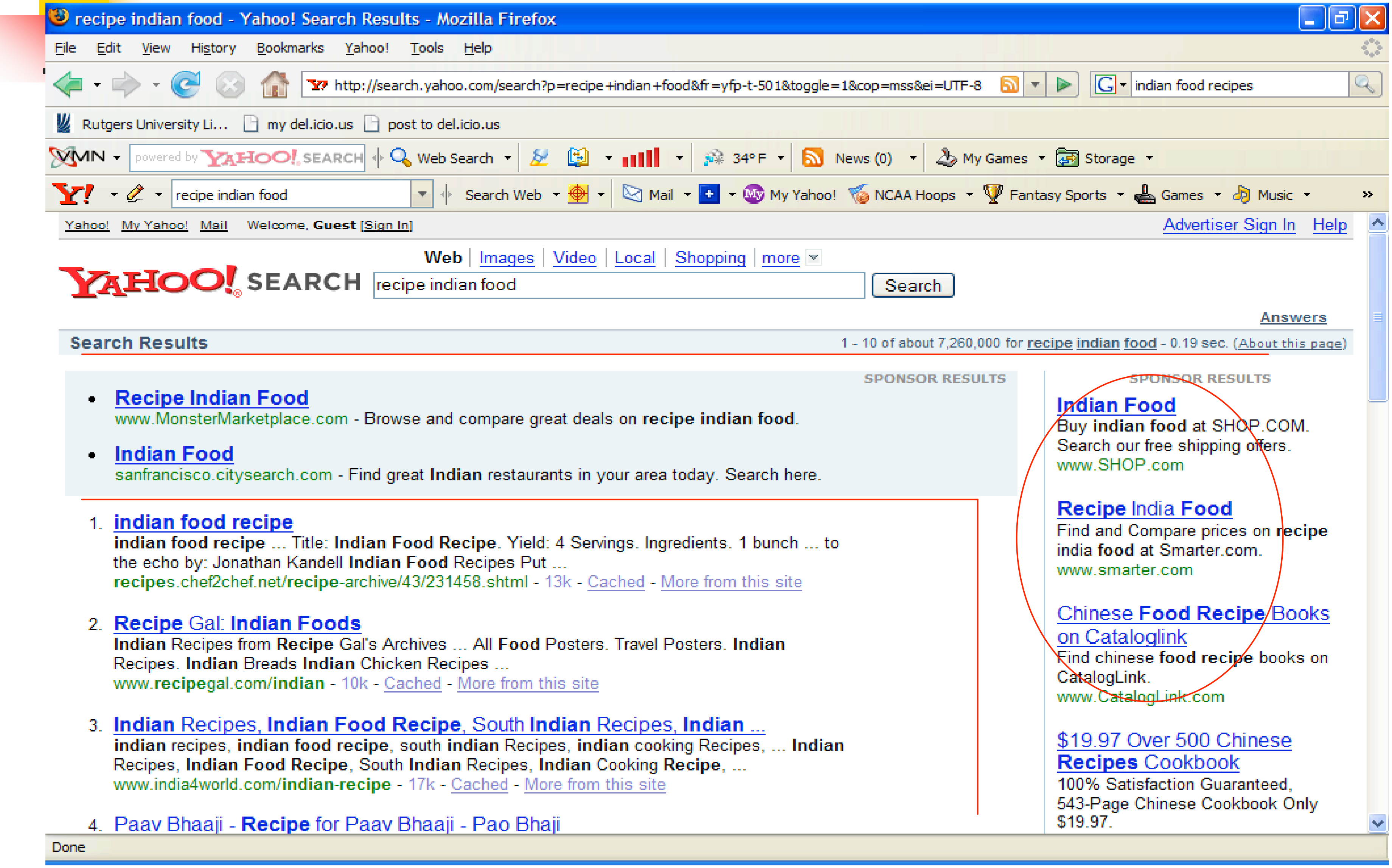


• Social nets	Nodes	Edges	Description
LiveJournal	4,843,953	42,845,684	Blog friendships [4]
Epinions	75,877	405,739	Who-trusts-whom [35]
Flickr	404,733	2,110,078	Photo sharing [21]
Delicious	147,567	301,921	Collaborative tagging
CA-DBLP	317,080	1,049,866	Co-authorship (CA) [4]
CA-cond-mat	21,363	91,286	CA cond-mat [25]
• Information networks			
Cit-hep-th	27,400	352,021	hep-th citations [13]
Blog-Posts	437,305	565,072	Blog post links [28]
• Web graphs			
Web-google	855,802	4,291,352	Web graph Google
Web-wt10g	1,458,316	6,225,033	TREC WT10G web
• Bipartite affiliation (authors-to-papers) networks			
AtP-DBLP	615,678	944,456	DBLP [25]
AtP-astro-ph	54,498	131,123	Arxiv astro-ph [25]
• Internet networks			
AS	6,474	12,572	Autonomous systems
Gnutella	62,561	147,878	P2P network [36]

Table 1: Some of the network datasets we studied.

Sponsored ("paid") Search

Text-based ads driven by user query


recipe indian food - Yahoo! Search Results - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://search.yahoo.com/search?p=recipe+indian+food&fr=yfp-t-501&toggle=1&cop=mss&ei=UTF-8
indian food recipes
Rutgers University Li... my del.icio.us post to del.icio.us
MSN powered by YAHOO! SEARCH Web Search 34° F News (0) My Games Storage
Y! recipe indian food Search Web Mail My Yahoo! NCAA Hoops Fantasy Sports Games Music
Yahoo! My Yahoo! Mail Welcome, Guest [Sign In] Advertiser Sign In Help
Web Images Video Local Shopping more
YAHOO! SEARCH recipe indian food Search
Answers
Done

Search Results 1 - 10 of about 7,260,000 for recipe indian food - 0.19 sec. (About this page)

SPONSOR RESULTS

- **Recipe Indian Food**
  www.MonsterMarketplace.com - Browse and compare great deals on **recipe indian food**.
- **Indian Food**
  sanfrancisco.citysearch.com - Find great **Indian** restaurants in your area today. Search here.

1. **indian food recipe**
   indian food recipe ... Title: **Indian Food Recipe**. Yield: 4 Servings. Ingredients. 1 bunch ... to the echo by: Jonathan Kandell **Indian Food** Recipes Put ...
   recipes.chef2chef.net/**recipe**-archive/43/231458.shtml - 13k - Cached - More from this site
2. **Recipe** Gal: **Indian Foods**
   Indian Recipes from **Recipe** Gal's Archives ... All **Food** Posters. Travel Posters. **Indian** Recipes. **Indian** Breads **Indian** Chicken Recipes ...
   www.**recipe**gal.com/**indian** - 10k - Cached - More from this site
3. **Indian** Recipes, **Indian Food Recipe**, South **Indian** Recipes, **Indian** ...
   indian recipes, **indian food recipe**, south **indian** Recipes, **indian** cooking Recipes, ... **Indian** Recipes, **Indian Food Recipe**, South **Indian** Recipes, **Indian** Cooking **Recipe**, ...
   www.india4world.com/**indian-recipe** - 17k - Cached - More from this site
4. Paav Bhaaji - **Recipe** for Paav Bhaaji - Pao Bhaji

SPONSOR RESULTS

Indian Food
Buy **indian food** at SHOP.COM. Search our free shipping offers.
www.SHOP.com

Recipe India **Food**
Find and Compare prices on **recipe** india **food** at Smarter.com.
www.smarter.com

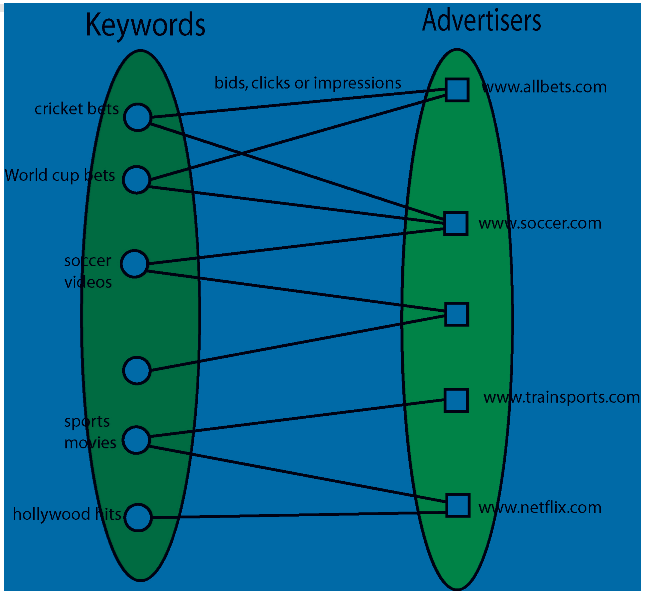
Chinese **Food Recipe** Books on Catalogink
Find chinese **food recipe** books on CatalogLink.
www.CatalogLink.com

$19.97 Over 500 Chinese **Recipes** Cookbook
100% Satisfaction Guaranteed, 543-Page Chinese Cookbook Only $19.97.

Sponsored Search Problems

Keyword-advertiser graph:

- provide new ads
- maximize CTR, RPS, advertiser ROI



Motivating cluster-related problems:

- Marketplace depth broadening:

  find new advertisers for a particular query/submarket

- Query recommender system:

  suggest to advertisers new queries that have high probability of clicks

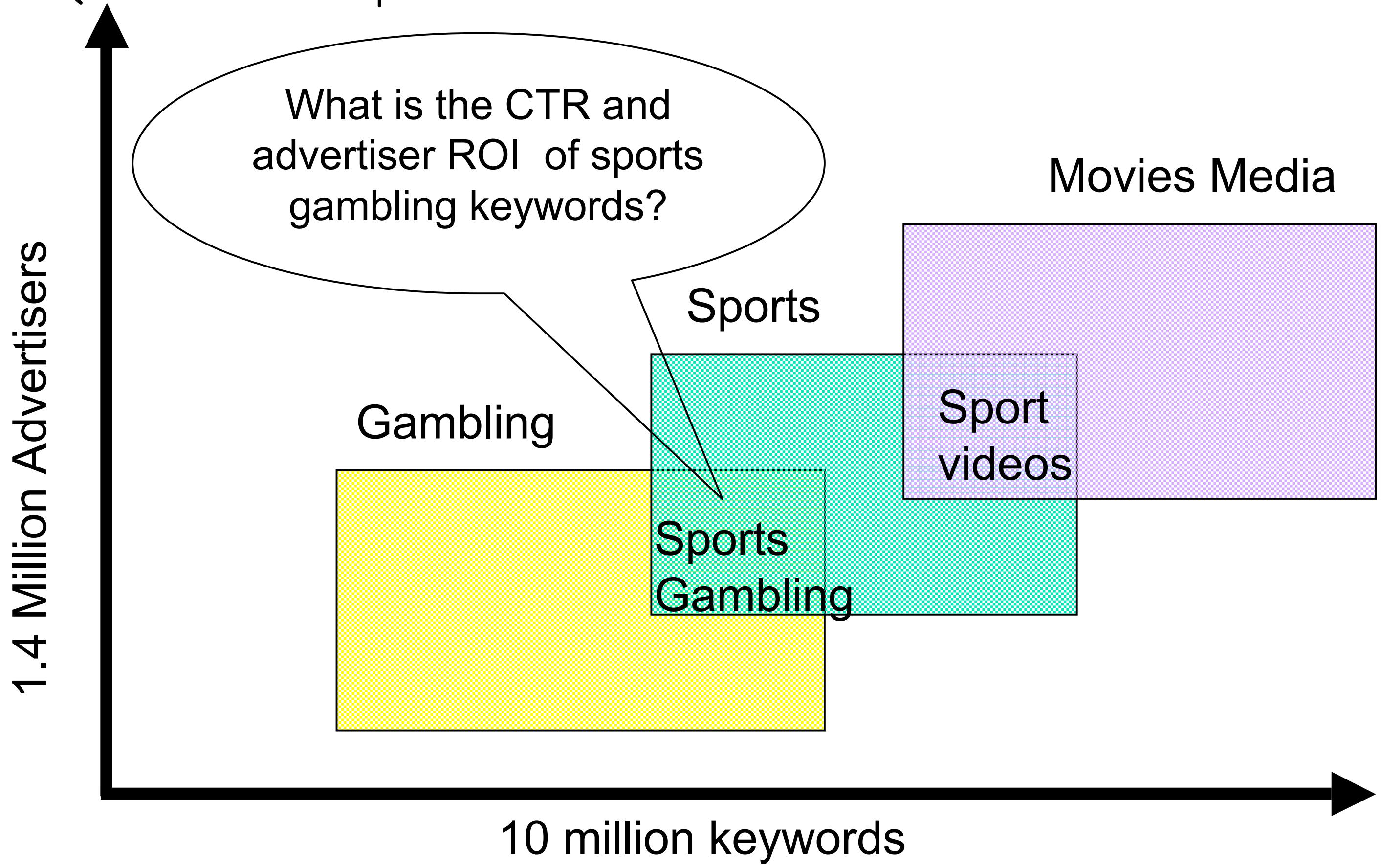
- Contextual query broadening:

  broaden the user's query using other context information

Micro-markets in sponsored search

Goal: Find *isolated* markets/clusters (in an advertiser-bidded phrase bipartite graph) with *sufficient money/clicks* with *sufficient coherence.*

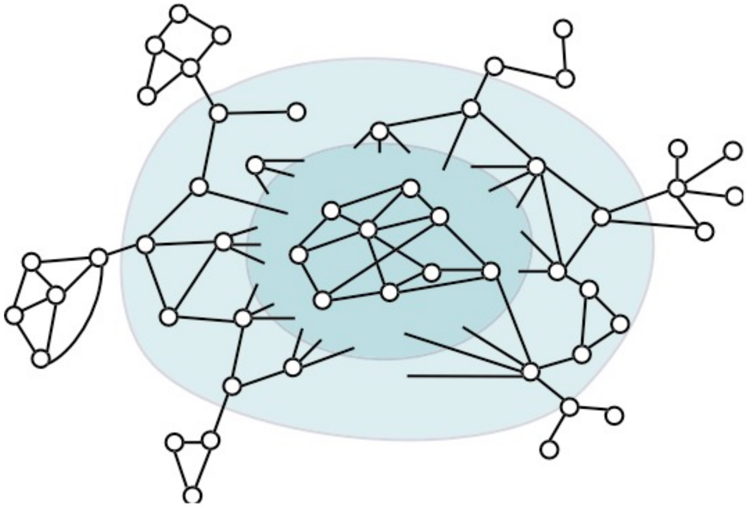
Ques: Is this even possible?



How people think about networks

"Interaction graph" ***model*** of networks:

- Nodes represent "entities"
- Edges represent "interaction" between pairs of entities

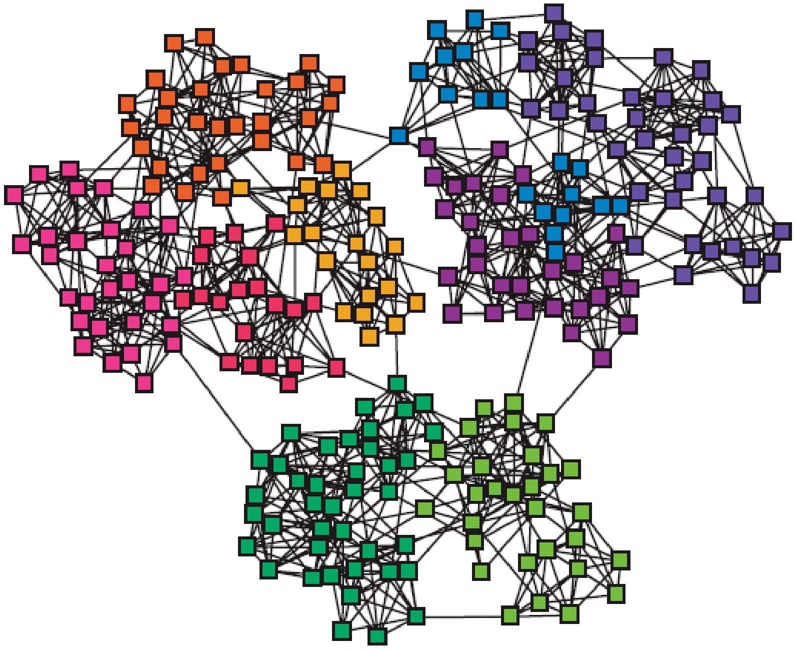


Graphs are combinatorial, not obviously-geometric

- Strength: powerful framework for analyzing *algorithmic complexity*
- Drawback: geometry used for learning and *statistical inference*

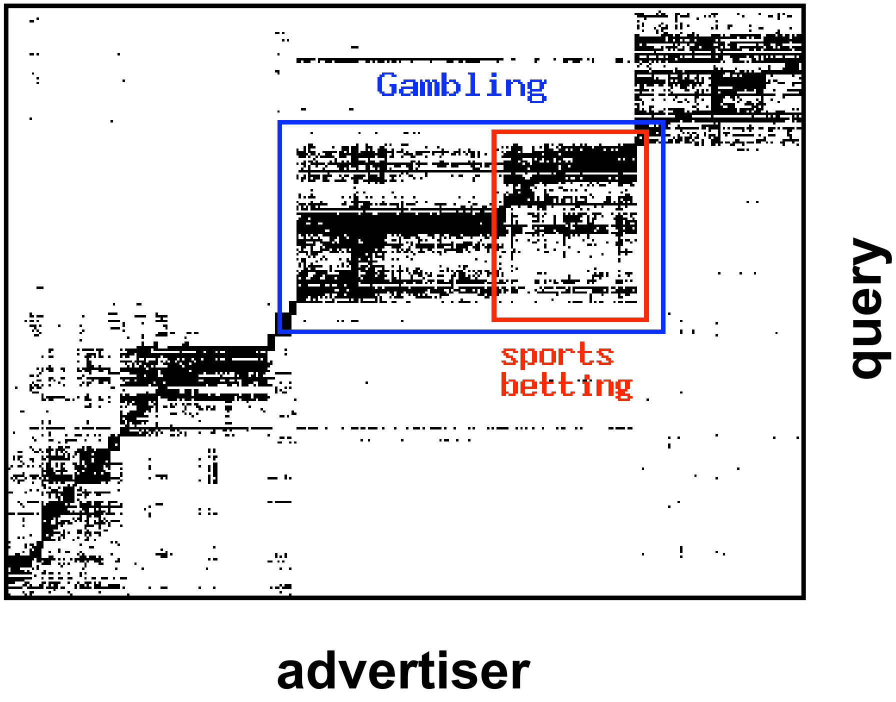
How people think about networks

A schematic illustration …



… of hierarchical clusters?

Some evidence for micro-markets in sponsored search?



Questions of interest ...

What are degree distributions, clustering coefficients, diameters, etc.?

Heavy-tailed, small-world, expander, geometry+rewiring, local-global decompositions, ...

Are there natural clusters, communities, partitions, etc.?

Concept-based clusters, link-based clusters, density-based clusters, ...

(e.g., *isolated* micro-markets with *sufficient money/clicks* with *sufficient coherence*)

How do networks grow, evolve, respond to perturbations, etc.?

Preferential attachment, copying, HOT, shrinking diameters, ...

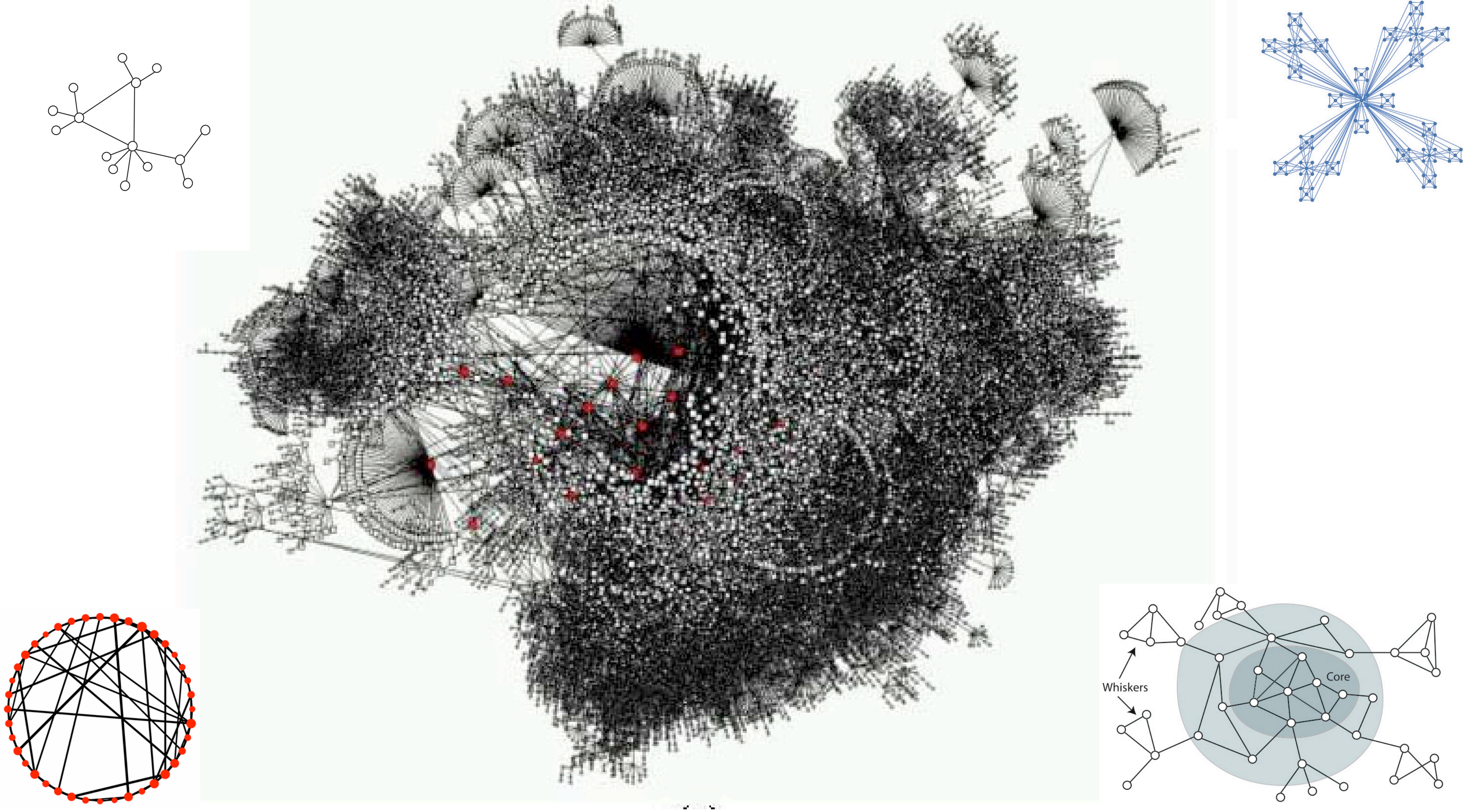
How do dynamic processes - search, diffusion, etc. - behave on networks?

Decentralized search, undirected diffusion, cascading epidemics, ...

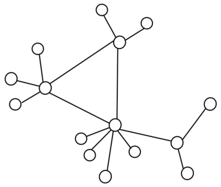
How best to do learning, e.g., classification, regression, ranking, etc.?

Information retrieval, machine learning, ...

What do these networks "look" like?



Popular approaches to large network data

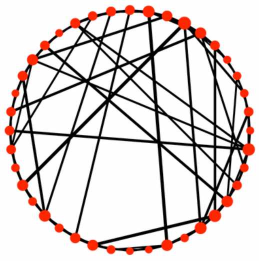


Heavy-tails and power laws (at *large size-scales*):

• extreme heterogeneity in local environments, e.g., as captured by degree distribution, and relatively unstructured otherwise

• basis for preferential attachment models, optimization-based models, power-law random graphs, etc.

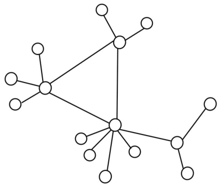
Local clustering/structure (at *small size-scales*):



• local environments of nodes have structure, e.g., captures with clustering coefficient, that is meaningfully "geometric"

• basis for small world models that start with global "geometry" and add random edges to get small diameter and preserve local "geometry"

Popular approaches to data more generally

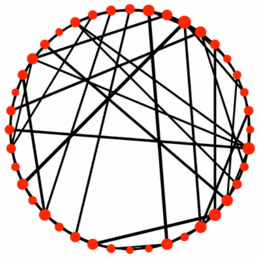


Use **geometric data analysis** tools:

- Low-rank methods - very popular and flexible
- Manifold methods - use other distances, e.g., diffusions or nearest neighbors, to find “curved” low-dimensional spaces

These geometric data analysis tools:

- View data as a point cloud in R^n, i.e., *each of the m data points is a vector in R^n*
- Based on SVD*, a basic vector space *structural* result
- Geometry gives a lot -- scalability, robustness, capacity control, basis for inference, etc.



*perhaps implicitly in an infinite-dimensional non-linearly transformed feature space (as with manifold and other Reproducing Kernel methods)

Can these approaches be combined?

These approaches are *very* different:

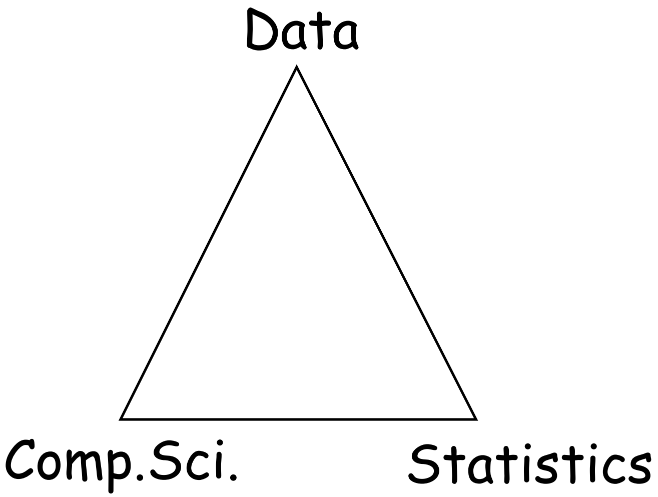
- network is a *single data point*---*not a collection of feature vectors* drawn from a distribution, and not really a matrix
- can't easily let m or n (number of data points or features) go to infinity---so nearly every such theorem fails to apply

Can associate matrix with a graph and vice versa, but:

- often do more damage than good
- questions asked tend to be *very different*
- graphs are really *combinatorial* things*

*But graph geodesic distance is a metric, and metric embeddings give fast algorithms!

Modeling data as matrices and graphs



In computer science:

- data are typically discrete, e.g., graphs
- focus is on fast algorithms for the given data set

In statistics*:

- data are typically continuous, e.g. vectors
- focus is on inferring something about the world

*very broadly-defined!

Algorithmic vs. Statistical Perspectives

Lambert (2000)

Computer Scientists

- Data: are a record of everything that happened.
- Goal: process the data to find interesting patterns and associations.
- Methodology: Develop approximation algorithms under different models of data access since the goal is typically computationally hard.

Statisticians

- Data: are a particular random instantiation of an underlying process describing unobserved patterns in the world.
- Goal: is to extract information about the world from noisy data.
- Methodology: Make inferences (perhaps about unseen events) by positing a model that describes the random variability of the data around the deterministic model.

Perspectives are NOT incompatible

- Statistical/probabilistic ideas are central to recent work on developing improved randomized algorithms for matrix problems.

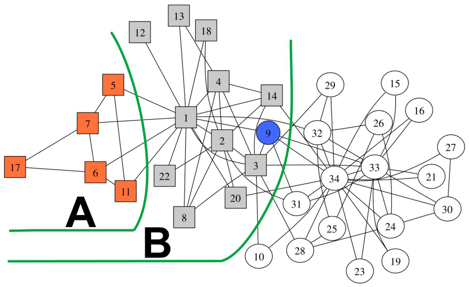
- Intractable optimization problems on graphs/networks yield to approximation when assumptions made about network participants.

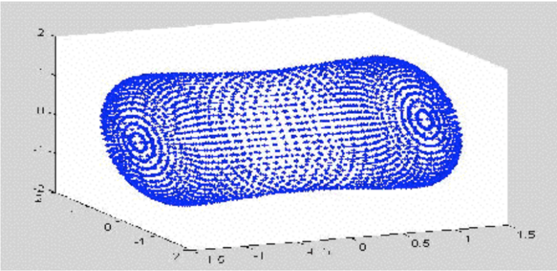
- In boosting, the computation parameter (i.e., the number of iterations) also serves as a regularization parameter.

- Approximations algorithms can implicitly regularize large graph problems (which can lead to *geometric network analysis tools*!).

What do the data "look like" (if you *squint* at them)?

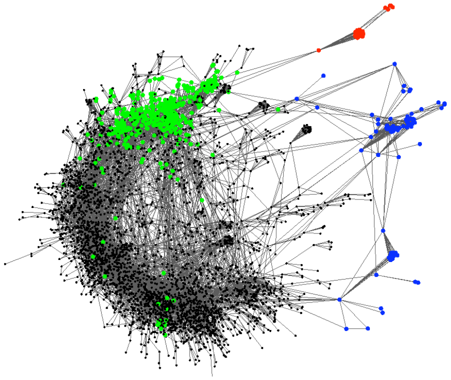
A "hot dog"?

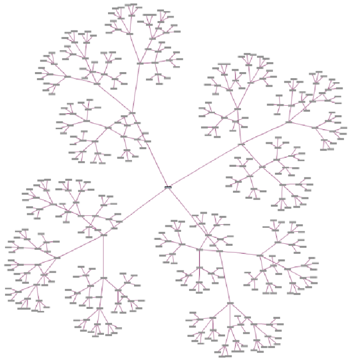




(or pancake that embeds well in low dimensions)

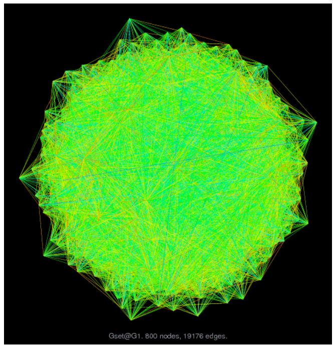
A "tree"?

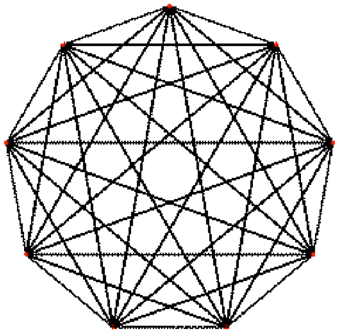




(or tree-like hyperbolic structure)

A "point"?





(or clique-like or expander-like structure)

Goal of the tutorial

Cover algorithmic and statistical work on identifying and exploiting "geometric" structure in large "networks"

- Address underlying theory, bridging the theory-practice gap, empirical observations, and future directions

Themes to keep in mind:

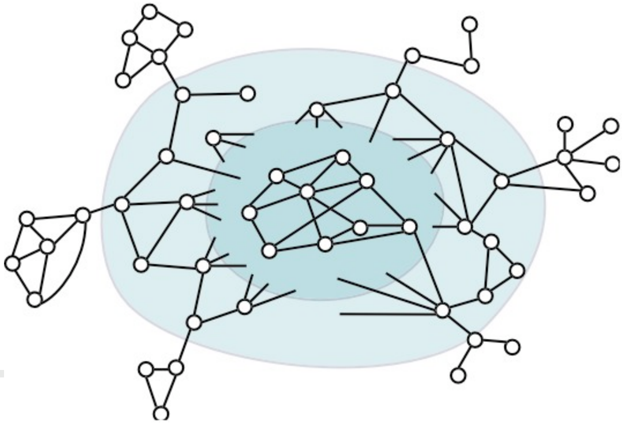
- Even infinite-dimensional Euclidean structure is too limiting

  (in adversarial environments, you never "flesh out" the low-dimensional space)

- Scalability and robustness are central

  (tools that do well on small data often do worse on large data)

Overview



Popular algorithmic tools with a geometric flavor

- PCA, SVD; interpretations, kernel-based extensions; algorithmic and statistical issues; and limitations

Graph algorithms and their geometric underpinnings

- Spectral, flow, multi-resolution algorithms; their implicit geometric basis; global and scalable local methods; expander-like, tree-like, and hyperbolic structure

Novel insights on structure in large informatics graphs

- Successes and failures of existing models; empirical results, including “experimental” methodologies for probing network structure, taking into account algorithmic and statistical issues; implications and future directions

Overview (more detail, 1 of 4)

Popular algorithmic tools with a geometric flavor

- PCA and SVD, including computational/algorithmic and statistical/geometric issues
- Domain-specific interpretation of spectral concepts, e.g., localization, homophily, centrality
- Kernel-based extensions currently popular in machine learning
- Difficulties and limitations of popular tools

Overview (more detail, 2 of 4)

Graph algorithms and their geometric underpinnings

- Spectral, flow, multi-resolution algorithms for graph partitioning, including theoretical basis and implementation issues

- Geometric and statistical perspectives, including "worst case" examples for each and behavior on "typical" classes of graphs

- Recent "local" methods and "cut improvement" methods; methods that "interpolate" between spectral and flow

- Tools for identifying "tree-like" or "hyperbolic" structure, and intuitions associated with this structure

Overview (more detail, 3 of 4)

Novel insights on structure in large informatics graphs

- Small-world and heavy-tailed models to capture local clustering and/or large-scale heterogeneity

- Issues of "pre-existing" versus "generated" geometry

- Empirical successes and failings of popular models, including densification, diameters, clustering, and community structure

- "Experimental" methodologies for "probing" network structure

Overview (more detail, 4 of 4)

Novel insights, (cont.)

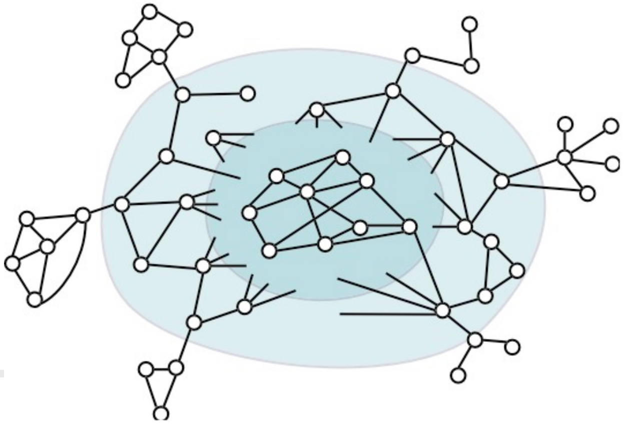
- Empirical results on "local" geometric structure, "global" metric structure, and the coupling between these

- Implicit regularization by worst-case approximation algorithms

- Implications for clustering, routing, information diffusion, visualization, and the design of machine learning tools

- Implications for dynamics evolution *of* graphs, dynamics *on* graphs, and machine learning and data analysis on networks

Overview



Popular algorithmic tools with a geometric flavor

- PCA, SVD; interpretations, kernel-based extensions; algorithmic and statistical issues; and limitations

Graph algorithms and their geometric underpinnings

- Spectral, flow, multi-resolution algorithms; their implicit geometric basis; global and scalable local methods; expander-like, tree-like, and hyperbolic structure

Novel insights on structure in large informatics graphs

- Successes and failures of existing models; empirical results, including "experimental" methodologies for probing network structure, taking into account algorithmic and statistical issues; implications and future directions

The Singular Value Decomposition (SVD)

The formal definition:

Given any m x n matrix A, one can decompose it as:

$$\underset{m \times n}{\begin{pmatrix} & & \\ & A & \\ & & \end{pmatrix}} = \underset{m \times \rho}{\begin{pmatrix} & & \\ & U & \\ & & \end{pmatrix}} \cdot \underset{\rho \times \rho}{\begin{pmatrix} \Sigma \end{pmatrix}} \cdot \underset{\rho \times n}{\begin{pmatrix} & V & \end{pmatrix}}^T$$

ρ: rank of A

U (V): orthogonal matrix containing the left (right) singular vectors of A.

Σ: diagonal matrix containing $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_\rho$, the singular values of A.

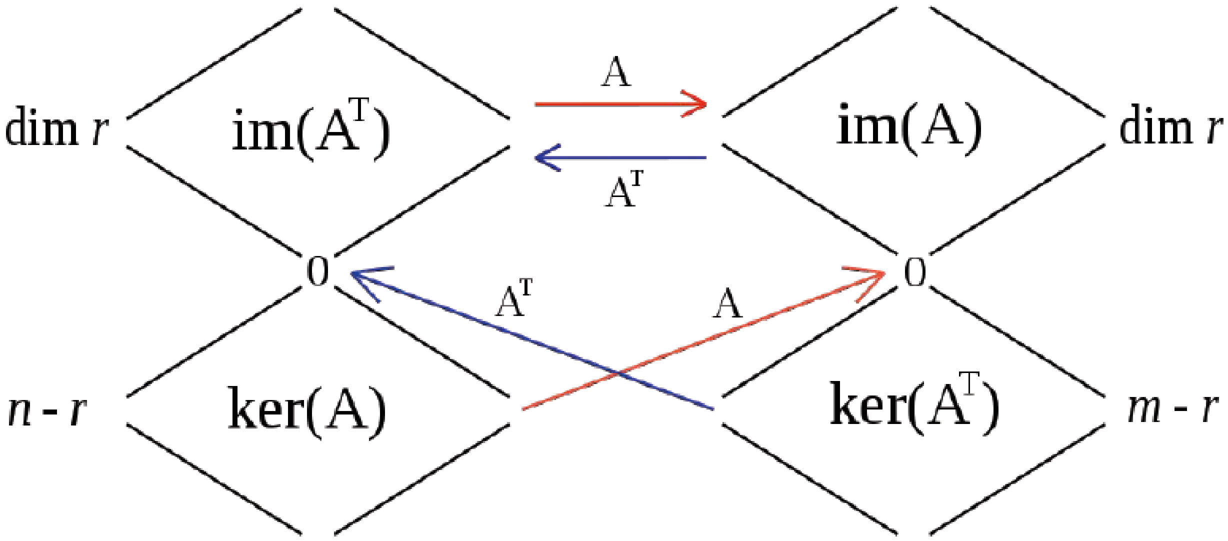
SVD is the "the Rolls-Royce and the Swiss Army Knife of Numerical Linear Algebra."*

*Dianne O'Leary, MMDS 2006

SVD: A fundamental *structural* result

SVD: a fundamental *structural* result of vector spaces (with both algorithmic and statistical consequences)

$$\begin{tikzcd}
\mathbb{R}^n \arrow[r, bend left, "A"] & \mathbb{R}^m \arrow[l, bend left, "A^T"]
\end{tikzcd}$$



U: orthogonal basis for the column space

V: orthogonal basis for the row space

Σ: gives *orthogonalized* "stretch" factors*

*i.e., in the basis of U and V, A is diagonal.

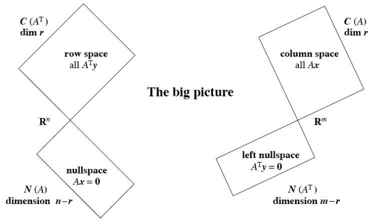


Figure 3.5: The dimensions of the Four Fundamental Subspaces (for R and for A).

Rank-*k* approximations (A_k)

Truncate the SVD at the top-k terms:

$$\underset{m \times n}{\begin{pmatrix} & & \\ & A_k & \\ & & \end{pmatrix}} = \underset{m \times k}{\begin{pmatrix} & & \\ & U_k & \\ & & \end{pmatrix}} \cdot \underset{k \times k}{\begin{pmatrix} & \Sigma_k & \end{pmatrix}} \cdot \underset{k \times n}{\begin{pmatrix} & & V_k^T & & \end{pmatrix}}$$

Keep the "most important" k-dim subspace.

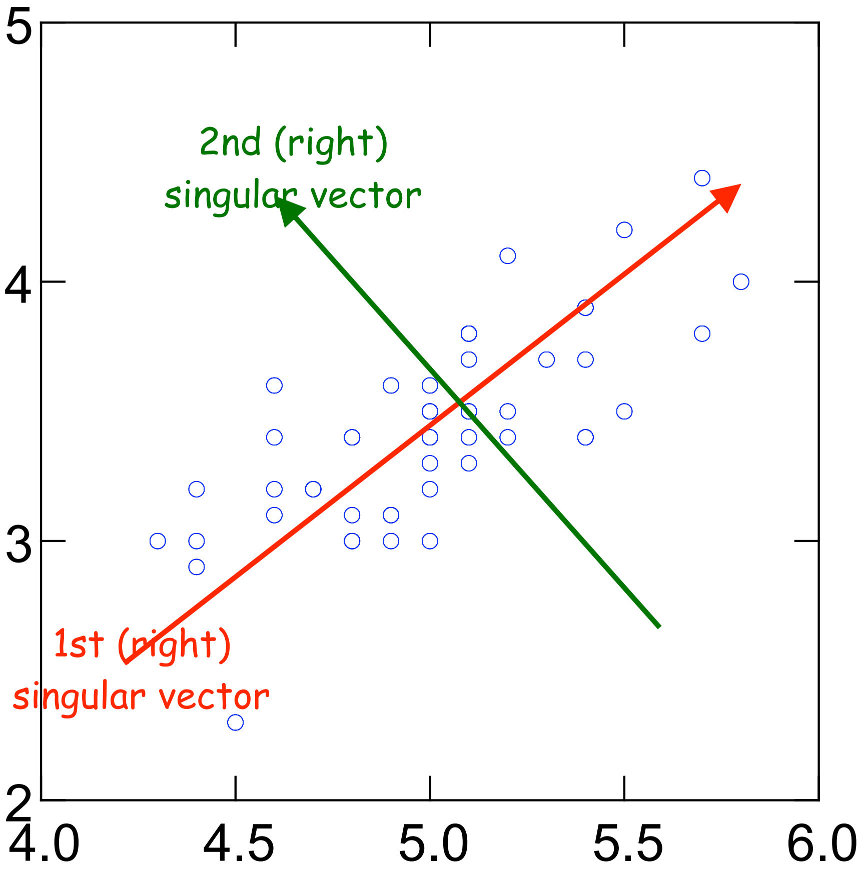
U_k (V_k): orthogonal matrix containing the top *k* left (right) singular vectors of *A*.

Σ_k: diagonal matrix containing the top *k* singular values of *A*.

Important: Keeping top k singular vectors provides "best" rank-k approximation to *A* (w.r.t. Frobenius norm, spectral norm, etc.):

$$A_k = \mathrm{argmin}\{ \, ||A-X||_{2,F} : \mathrm{rank}(X) \leq k \, \}.$$

Singular vectors, intuition



Let the blue circles represent *m* data points in a 2-D Euclidean space.

Then, the SVD of the *m-by-2* matrix of the data will return …

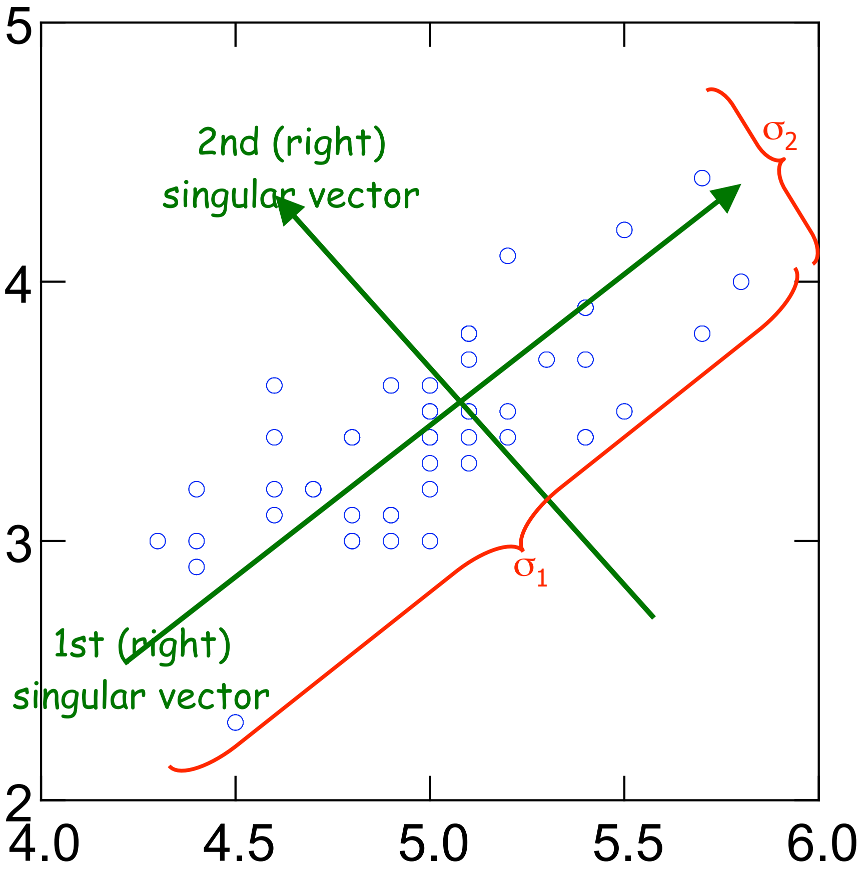
1st (right) singular vector:

direction of maximal variance,

2nd (right) singular vector:

direction of maximal variance, after removing the projection of the data along the first singular vector.

Singular values, intuition



σ_1: measures how much of the data variance is explained by the first singular vector.

σ_2: measures how much of the data variance is explained by the second singular vector.

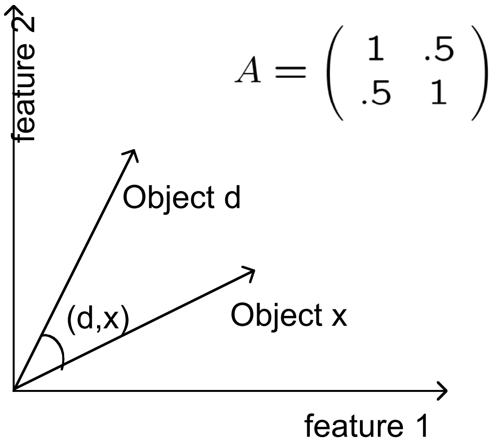
A first use of the SVD in data analysis

Common to *model the data* as points in a vector space -- this gives a matrix, with *m* rows (one for each object) and *n* columns (one for each feature).

Matrix rows: points (vectors) in a Euclidean space, e.g., given 2 objects (*x* & *d*), each described with respect to two features, we get a 2-by-2 matrix.

Common assumption: Two objects are "close" if angle between their corresponding vectors is "small."

Common hope: k « m,n directions are important -- e.g., A_k captures most of the "information" and/or is "discriminative" for classification, etc tasks.



$$A = \begin{pmatrix} 1 & .5 \\ .5 & 1 \end{pmatrix}$$

LSI: A_k for document-term “matrices”

(Berry, Dumais, and O'Brien '92)

Latent Semantic Indexing (LSI)

Replace A by A_k; apply clustering/classification algorithms on A_k.

$$\begin{array}{cc} & n \text{ terms (words)} \\ \begin{array}{c} m \\ \text{documents} \end{array} & \left(\begin{array}{c} \mathcal{A} \\ \\ A_{ij} = \text{frequency of j-th} \\ \text{term in i-th document} \end{array}\right) \end{array}$$

Pros

- Less storage for small *k*.

O(km+kn) vs. O(mn)

- Improved performance.

Documents are represented in a “concept” space.

Cons

- A_k destroys sparsity.

- Interpretation is difficult.

- Choosing a good *k* is tough.

- Sometimes people interpret document corpus in terms of k topics when use this.
- Better to think of this as just selecting one model from a parameterized class of models!

LSI/SVD and heavy-tailed data

Theorem: (Mihail and Papadimitriou, 2002)

The largest eigenvalues of the adjacency matrix of a graph with power-law distributed degrees are also power-law distributed.

- I.e., heterogeneity (e.g., heavy-tails over degrees) plus noise (e.g., random graph) implies heavy tail over eigenvalues.
- Idea: 10 components may give 10% of mass/information, but to get 20%, you need 100, and to get 30% you need 1000, etc; i.e., no scale at which you get most of the information
- No "latent" semantics without preprocessing.

Singular-stuff and eigen-stuff

If A is any m x n matrix:

$A = U \Sigma V^T$ (the SVD - general eigen-systems can be non-robust and hard to work with)

A is diagonal in orthogonal U and V basis; and Σ nonnegative

If A is any m x m square matrix:

$A = U \Lambda U^T$ (the eigen-decomposition - of course, A also has an SVD)

A is diagonal in orthogonal U basis; but Λ is not nonnegative

If A is any m x m SPSD (i.e., correlation) matrix:

$A = U \Sigma U^T$ (SVD = eigen-decomposition)

A is diagonal in orthogonal U basis; and Σ nonnegative

In data analysis, structural properties of SVD are used most often via square (e.g., adjacency) or SPSD (e.g., kernel or Laplacian) matrices

Algorithmic Issues with the SVD

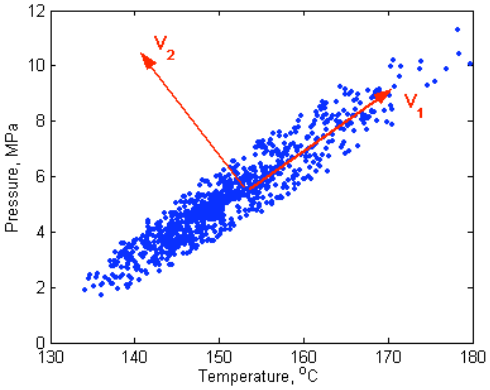
A *big* area with a lot of subtleties:

- "Exact" computation of the *full SVD** takes $O(\min\{mn^2, m^2n\})$ time.
- The top *k* left/right singular vectors/values can be computed faster using *iterative* Lanczos/Arnoldi methods.
- Specialized numerical methods for very large sparse matrices.
- A lot of work in TCS, NLA, etc on randomized algorithms and ε-approximation algorithms (for $\varepsilon \approx 0.1$ or $\varepsilon \approx 10^{-16}$).

*Given the full SVD, you can do "everything." But *you "never" need the full SVD*. Just compute what you need!

PCA and MDS

Principal Components Analysis (PCA)



- Given $\{X_i\}_{i=1,\dots,n}$ with $X_i \varepsilon R^D$,

  Find k-dimensional subspace P and embedding $Y_i=PX_i$

  s.t. Variance(Y) is maximized or Error(Y) is minimized

- Do SVD on covariance matrix $C = XX^T$

Multidimensional Scaling (MDS)

- Given $\{X_i\}_{i=1,\dots,n}$ with $X_i \varepsilon R^D$,

  Find k-dimensional subspace P and embedding $Y_i=PX_i$

  s.t. $\mathrm{Dist}(Y_i-Y_j) \approx \mathrm{Dist}(X_i-X_j)$, i.e., dot products (or distances) preserved

- Do SVD on Gram matrix $G = X^T X$

SVD is the structural basis behind PCA, MDS, Factor Analysis, etc.

Statistical Aspects of the SVD

Can always compute best rank-k SVD approximation

- in "nice" Gaussian settings, corresponding statistical interpretation
- more generally, model selection in a place with nice geometry

Least-squares regression and PCA

- optimal (in terms of mean squared error) linear compression scheme for compressing and reconstructing any high-dimensional vectors
- if the data were generated from Guassian distributions, then it is the "right thing to do"
- several related ways to formalize these ideas

Geometric Aspects of the SVD

Can always compute best rank-k SVD approximation

- in “nice” Gaussian settings, corresponding statistical interpretation
- more generally, model selection in a place with nice geometry

Least-squares regression and PCA

- embed the data in a line or low-dimensional hyperplane
- reconstruct clusters when data consist of “separated” Gaussians
- geometry permits Nystrom-based and other out-of-sample schemes and “robustness” due to constraints imposed by low-dimensional space
- several related ways to formalize these ideas

These are a *very* strong properties

Contrast these properties with tensors*

- Computing the rank of a tensor (*qua* tensor) is intractable, and best rank k approximation may not even exist

- Many other strong hardness results (Lim 2006)

- Researchers “fall back” on matrices along each mode

That matrices are so nice is the exception, not the rule, among algebraic structures---vector spaces are *very* structured places, with associated benefits and limitations.

*Tensors are another algebraic structure used to model data: Think of them as A_{ijk}, i.e., matrices with an additional subscript, where multiplication is linear along each “direction”

Kernel Methods

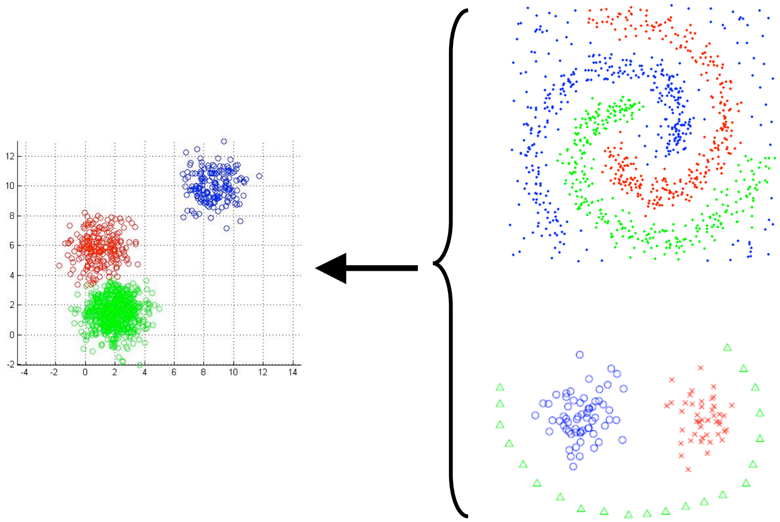
Many algorithms access data only through elements of Correlation or Gram matrix.

$$< x_i, x_j > \longrightarrow K(x_i, x_j)$$

- Can use another SPSD matrix and to encode nearness information.

- Many learning bounds generalize

- E.g., $K(x_i,x_j) = f(||x_i-x_j||)$, Gaussian r.b.f., polynomial kernels, etc - good but limited



- Data-dependent kernels - operationally define a kernel on graph constructed from point cloud data; typically viewed as implicitly defining a *manifold*

Kernels and linear methods

Kernel methods are basically linear methods in some other feature space that is non-linearly related to the original representation of the data:

- Good news: still linear (classify with hyperplanes, have capacity control since hyperplanes are structured objects, etc.)

- Bad news: still linear (so still boiling down to SVD); determining features is an art; very hard to deal with very non-linear metrics

Kernel methods basically give you a lot more statistical (or descriptive) flexibility without too much additional computational cost.

Data-dependent kernels, cont.

ISOMAP:

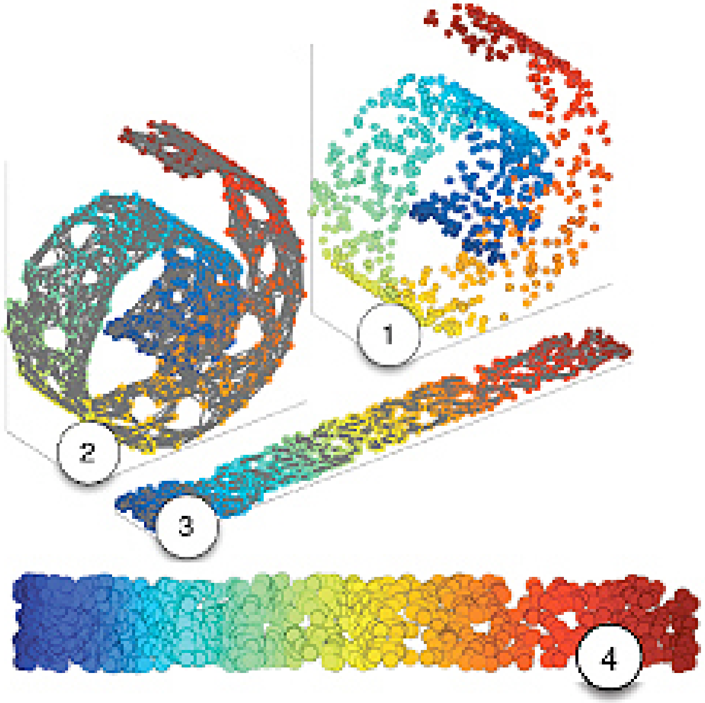
- Compute geodesics on adjacency graph
- MetricMDS gives k eigenvectors for embedding

LLE:

- Compute edge weights from local least-squares approximation
- Compute global embedding vectors as bottom k+1 eigenvectors of a matrix

Laplacian eigenmaps:

- Assign edge weights $W_{ij} = \exp(-\beta||x_i - x_j||_2^2)$
- Compute embedding vectors as bottom k+1 eigenvectors of Laplacian



Kernels and Manifolds and Diffusions

Laplacian Eigenmaps:

- Defined on graphs, but close connections to “analysis on manifolds”

Laplacian in R^d: $$\Delta f = -\sum_i \frac{\partial^2 f}{\partial^2 x_i}$$

Manifold Laplacian

- measure change along tangent space of manifold

Connections with diffusions (and Markov chains):

$$\frac{\partial \Psi(t)}{\partial t} = -L\Psi(t)$$

$$K_{(t)} = \exp(-Lt) \ (\sim \text{Green's function})$$

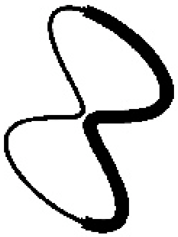
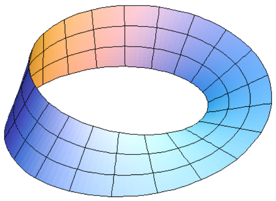
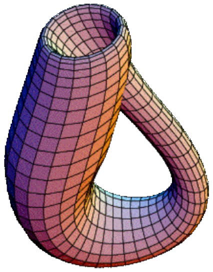
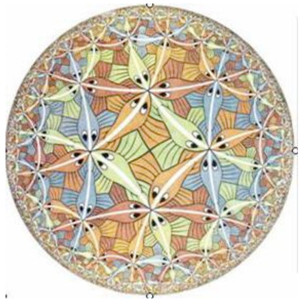
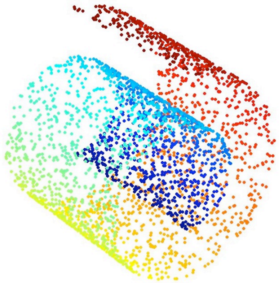
$$\Psi(t) = K_{(t)}\Psi(0)$$

$$K_{(t)} \sim \frac{1}{2}L^+ \ (\text{under “nice” sssumptions})$$

What is a manifold?

A topological manifold is a topological space which locally looks Euclidean in a certain (weak) sense

A Riemannian manifold is a *differentiable* manifold in which the tangent space is R^n. (Tangent space has *inner product* that varies smoothly and that gives lengths, angles, areas, gradients, etc.)

*Barring "pathological" curvature or density behavior, i.e., permitting a **huge** amount of descriptive flexibility, think of a ML manifolds as a "curved" low-dimensional space.*

Kernels and learning a manifold

Practice and Theory:

- Choose kernel, and see if eigen-methods give good visualization, clustering, etc.

- Thm: If the hypothesized manifold and sampling density are "nice," then L_{graph} will converge to $L_{manifold}$.

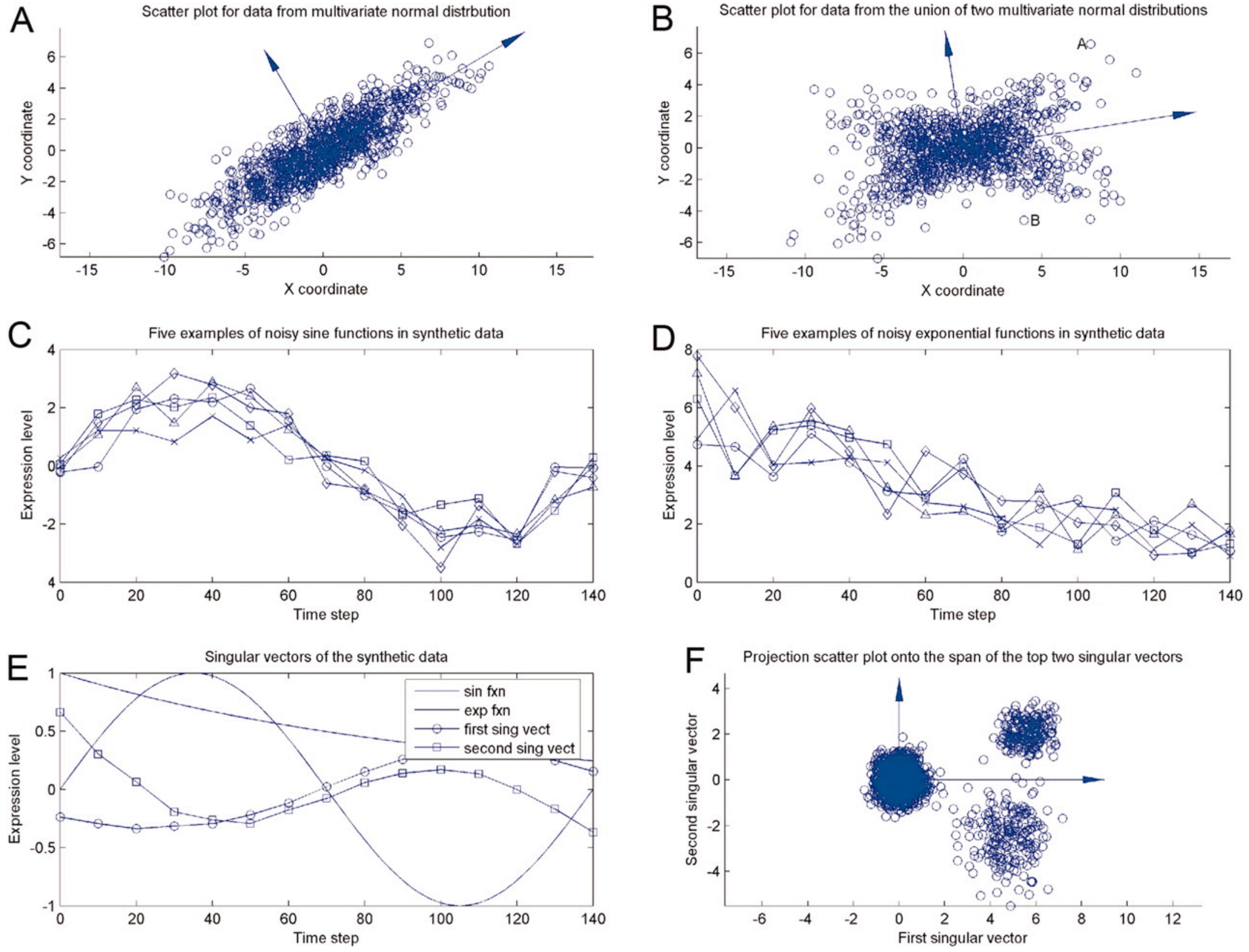
Manifold learning is *not* of classification, clustering, regression; but of the hypothesized manifold

- Empirically (or theoretically) useful when two large clusters

- Basically, "exploratory" data modeling, using one class of models

Interpreting the SVD - be *very* careful

Mahoney and Drineas (PNAS, 2009)

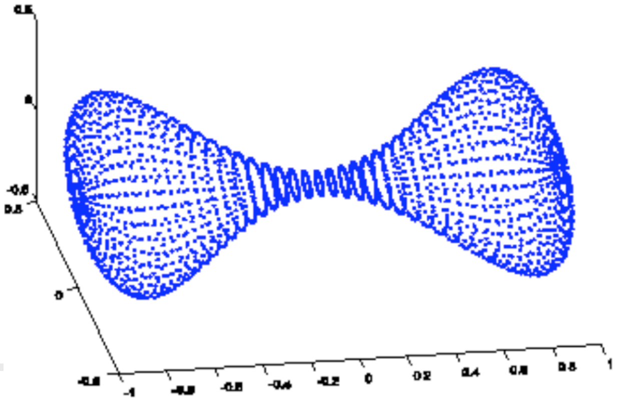


Reification

- assigning a “physical reality” to large singular directions
- invalid in general

Just because “If the data are ‘nice’ then SVD is appropriate” does NOT imply converse.

Interpretation: Centrality



Centrality (of a vertex) - measures relative importance of a vertices in a graph

- degree centrality - number of links incident upon a node
- betweenness centrality - high for vertices that occur on many shortest paths
- closeness centrality - mean geodesic distance between a vertex and other reachable nodes
- eigenvector centrality - connections to high-degree nodes are more important, and so on iteratively (a "spectral ranking" measure)

Motivation and behavior on nice graphs is clear -- but what do they actually compute on non-nice graphs?

Eigen-methods in ML and data analysis

Eigen-tools appear (*explicitly* or *implicitly**) in many data analysis and machine learning tools:

- Latent semantic indexing
- Manifold-based ML methods
- Diffusion-based methods
- k-means clustering
- Spectral partitioning and spectral ranking

*What are the limitations imposed when these methods are implicitly used? Can we get around those limitations with complementary methods?

k-means clustering

(Drineas, Frieze, Kannan, Vempala, and Vinay '99; Boutsidis, Mahoney, and Drineas '09)

k-means clustering

A standard objective function that measures cluster quality.

(Often denotes an iterative algorithm that attempts to optimize the *k*-means objective function.)

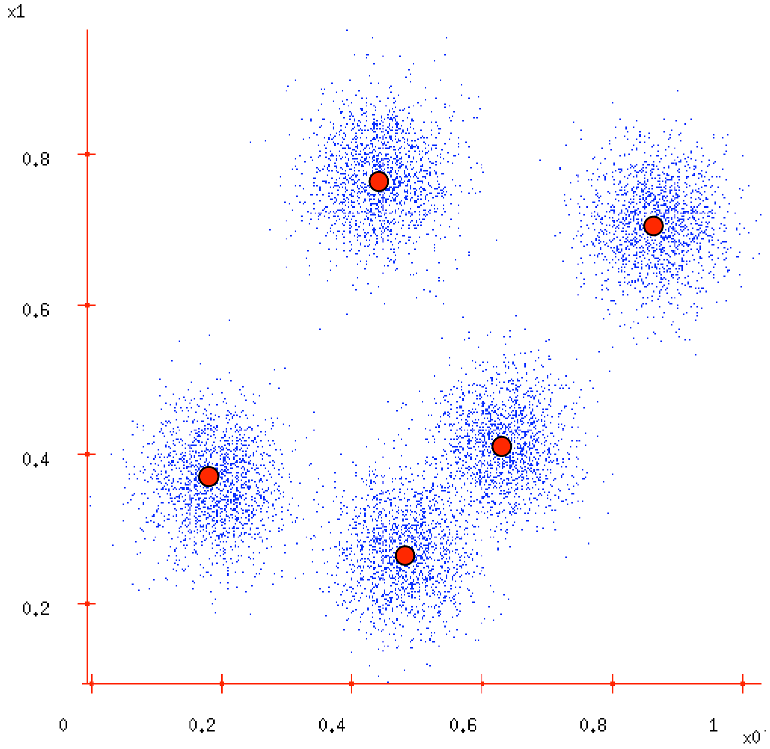
k-means objective

Input: set of m points in R^n, positive integer k

Output: a partition of the m points to k clusters

Partition the m points to k clusters in order to minimize the sum of the squared Euclidean distances from each point to its cluster centroid.

k-means clustering, cont'd



Goal: We seek to split the input points in 5 clusters.

Recall: The cluster centroid is the "average" of all the points in the cluster:

$$\operatorname{argmin}_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} ||\mathbf{x}_j - \mu_i||_2^2$$

Note: The *intuition* underlying the combinatorial objective is that there are several "nice" clusters in a low-dimensional space.

k-means: a matrix formulation

Let *A* be the *m-by-n* matrix representing *m* points in R^n. Then, we seek to

$$\min_{X \in \mathbb{R}^{m \times k}} \|A\|_F^2 - \left\|X^T A\right\|_F^2 \qquad \text{or} \qquad \max_{X \in \mathbb{R}^{m \times k}} \left\|X^T A\right\|_F^2$$

X is a special "cluster membership" matrix: X_{ij} denotes if the *i-th* point belongs to the *j-th* cluster.

$$\begin{array}{cc} & \text{clusters} \\ \text{points} & \begin{pmatrix} \\ \\ X \\ \\ \\ \end{pmatrix} \\ & m \times k \end{array}$$

- Columns of X are normalized to have unit length.

(We divide each column by the square root of the number of points in the cluster.)

- *Every row of X has at most one non-zero element.*

(Each element belongs to at most one cluster.)

- X is an orthogonal matrix, i.e., $X^TX = I$.

k-means: the SVD connection

If we only require that X is an orthogonal matrix and remove the condition on the number of non-zero entries per row of X, then

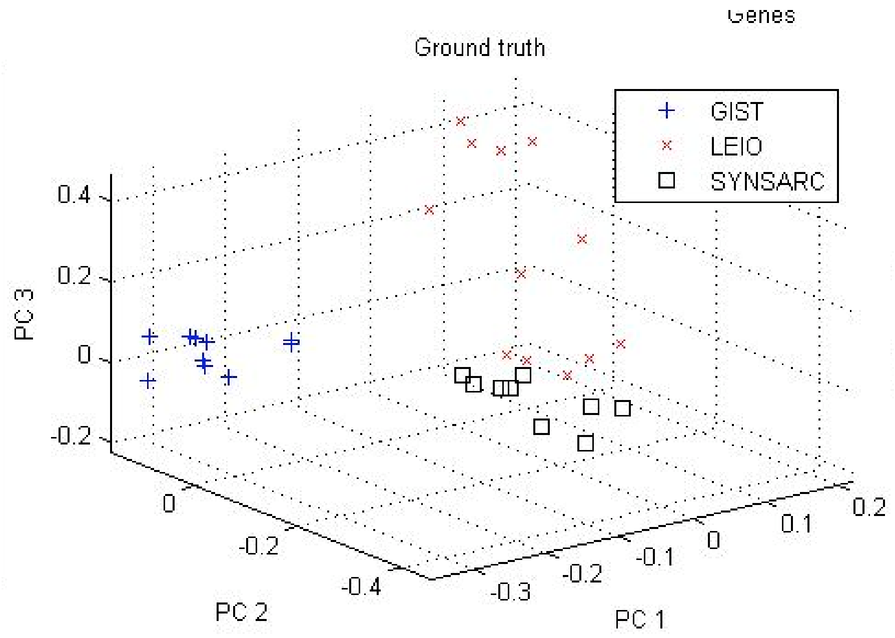
$$\min_{X \in \mathbb{R}^{m \times k}} \|A\|_F^2 - \left\|X^T A\right\|_F^2 \qquad \text{or} \qquad \max_{X \in \mathbb{R}^{m \times k}} \left\|X^T A\right\|_F^2$$

is easy to minimize! The solution is $X = U_k$.

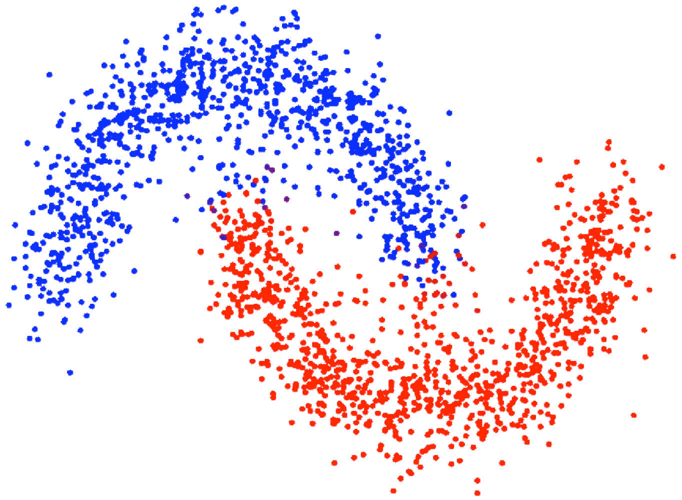
Using SVD to solve *k*-means

- We can get a 2-approximation algorithm for k-means. (Drineas, Frieze, Kannan, Vempala, and Vinay '99, '04)
- We can get heuristic schemes to assign points to clusters. (Zha, He, Ding, Simon, and Gu '01)
- There exist PTAS (based on random projections) for *k*-means problem. (Ostrovsky and Rabani '00, '02)
- Deeper connections between SVD and clustering. (Kannan, Vempala, and Vetta '00, '04)

k-means and “kernelized” k-means



Regular k-means in R^3



“Kernelized” k-means in some transformed space

A few high-level observations

Eigenvectors are global entities--awkward to find local structure.

- Basically, due to the orthogonality requirement -- usually, the most significant thing about the 17th eigenvector is that it is orthogonal to the first 16!
- Typically only the top few eigenvectors can be localized.

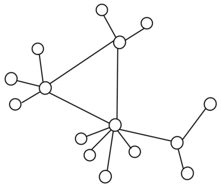
Eigenvectors identify linear structure

- Can associate matrix with any graph, but questions you ask are different -- e.g., what is the matrix that is least like a "low-dimensional" matrix?
- That is why we kernelize -- to be linear somewhere else and exploit eigen-methods.

Eigen-tools and the SVD give "sweet spot" between *descriptive flexibility* and *algorithmic tractability*

- E.g., analogue of SVD for tensors and other *algebraic* structures fails to hold -- so researchers there fall back on the SVD too.
- Question: Are there other "sweet spots" when eigen-methods are too limited?

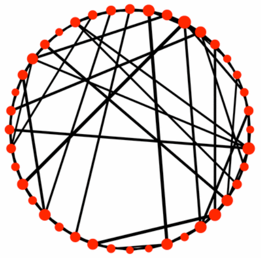
Unfortunately …



Relationship b/w small-scale and large-scale structure is not reproduced (even qualitatively) by popular models

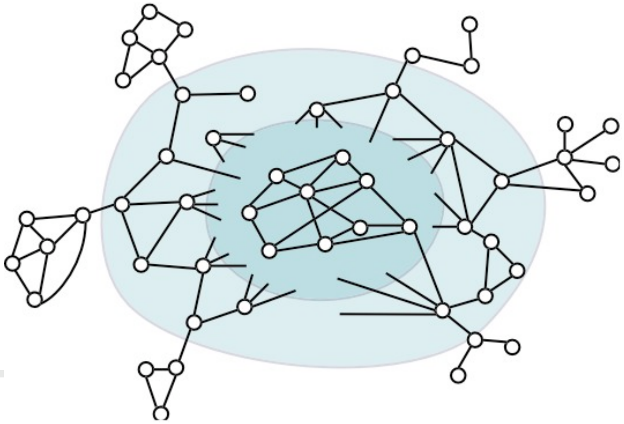
- Relationship governs diffusion of information; decentralized search; routing; dynamic properties; applicability of common ML tools

Also: ∃ a BIG disconnect b/w common data analysis tools and network properties



- low-dimensional & geometric tools (SVD, diffusion-based manifold methods, ...) common in ML, but networks are more expander-like
- *network is single data point*---not really a bunch of feature vectors

Overview



Popular algorithmic tools with a geometric flavor

- PCA, SVD; interpretations, kernel-based extensions; algorithmic and statistical issues; and limitations

Graph algorithms and their geometric underpinnings

- Spectral, flow, multi-resolution algorithms; their implicit geometric basis; global and scalable local methods; expander-like, tree-like, and hyperbolic structure

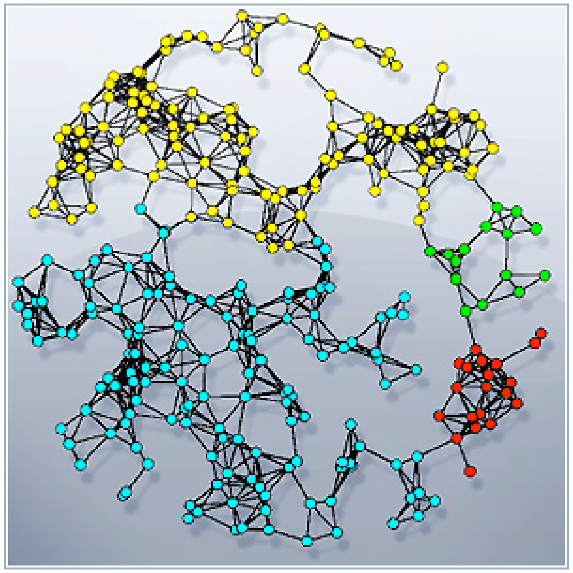
Novel insights on structure in large informatics graphs

- Successes and failures of existing models; empirical results, including “experimental” methodologies for probing network structure, taking into account algorithmic and statistical issues; implications and future directions

Graph partitioning

A family of combinatorial optimization problems - want to partition a graph's nodes into two sets s.t.:

- Not much edge weight across the cut (cut quality)
- Both sides contain a lot of nodes



Several standard formulations:

- Graph bisection (minimum cut with 50-50 balance)
- β-balanced bisection (minimum cut with 70-30 balance)
- cutsize/min{|A|,|B|}, or cutsize/(|A||B|) (expansion)
- cutsize/min{Vol(A),Vol(B)}, or cutsize/(Vol(A)Vol(B)) (conductance or N-Cuts)

All of these formalizations of the bi-criterion are NP-hard!

Why graph partitioning? (1 of 2*)

Graph partitioning algorithms:

- capture a qualitative notion of connectedness
- well-studied problem in traditionally/recently both in theory and practice
- many machine learning and data analysis applications

Don't care about exact solution to intractable problem:

- output of approximation algs is not something we "settle for"
- randomized/approximation algs often give "better" answers than exact solution
- nearly-linear/poly-time computation captures "qualitative existence"

*(2 of 2) is later

Squint at the data graph …

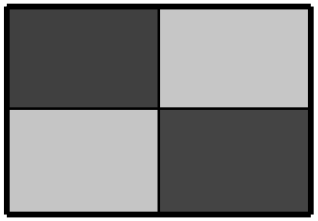
Say we want to find a "best fit" of the adjacency matrix to:

$$\begin{array}{|c|c|}\hline \alpha & \beta \\ \hline \beta & \gamma \\ \hline \end{array}$$

What does the data "look like"? How big are α, β, γ?

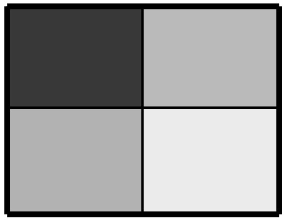
$\alpha \approx \gamma \gg \beta$

low-dimensional



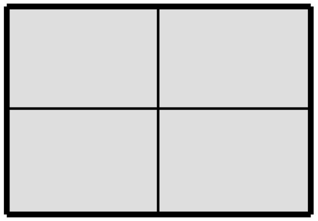
$\alpha \gg \beta \gg \gamma$

core-periphery



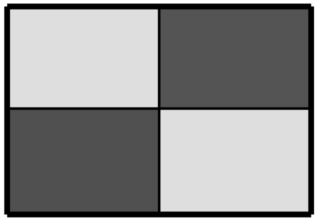
$\alpha \approx \beta \approx \gamma$

expander or K_n



$\beta \gg \alpha \approx \gamma$

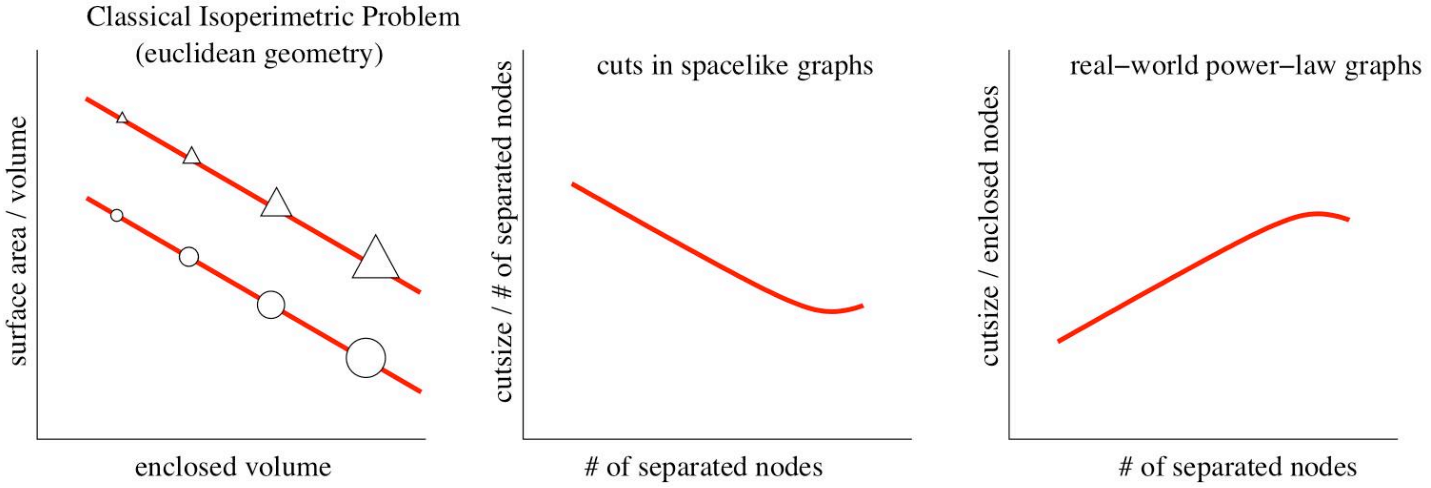
bipartite graph



Why worry about both criteria?

- Some graphs (e.g., "space-like" graphs, finite element meshes, road networks, random geometric graphs) cut quality and cut balance "work together"

Tradeoff between cut quality and balance



- For other classes of graphs (e.g., informatics graphs, as we will see) there is a "tradeoff," i.e., better cuts lead to worse balance
- For still other graphs (e.g., expanders) there are no good cuts of any size

The “lay of the land“

Spectral methods - compute eigenvectors of associated matrices

Local improvement - easily get trapped in local minima, but can be used to clean up other cuts

Multi-resolution - view (typically space-like graphs) at multiple size scales

Flow-based methods - single-commodity or multi-commodity version of max-flow-min-cut ideas

Spectral Methods

Fiedler (1973) and Donath & Hoffman (1973)

- use eigenvectors of discrete graph Laplacian

Popular in scientific computing, parallel computing, etc. (1980s) and machine learnint (200s)

Algorithm:

1. Compute the exact/approximate eigenvector.

2. Perform “rounding”: choose the best of the n cuts defined by that eigenvector.

Cheeger's inequality

Theorem: If $\lambda_2(g)$ is second eigenvalue of Laplacian and $\phi(G)$ is the conductance, then

$$\lambda_2(G)/2 \leq \phi(G) \leq \sqrt{8\lambda_2(G)}$$

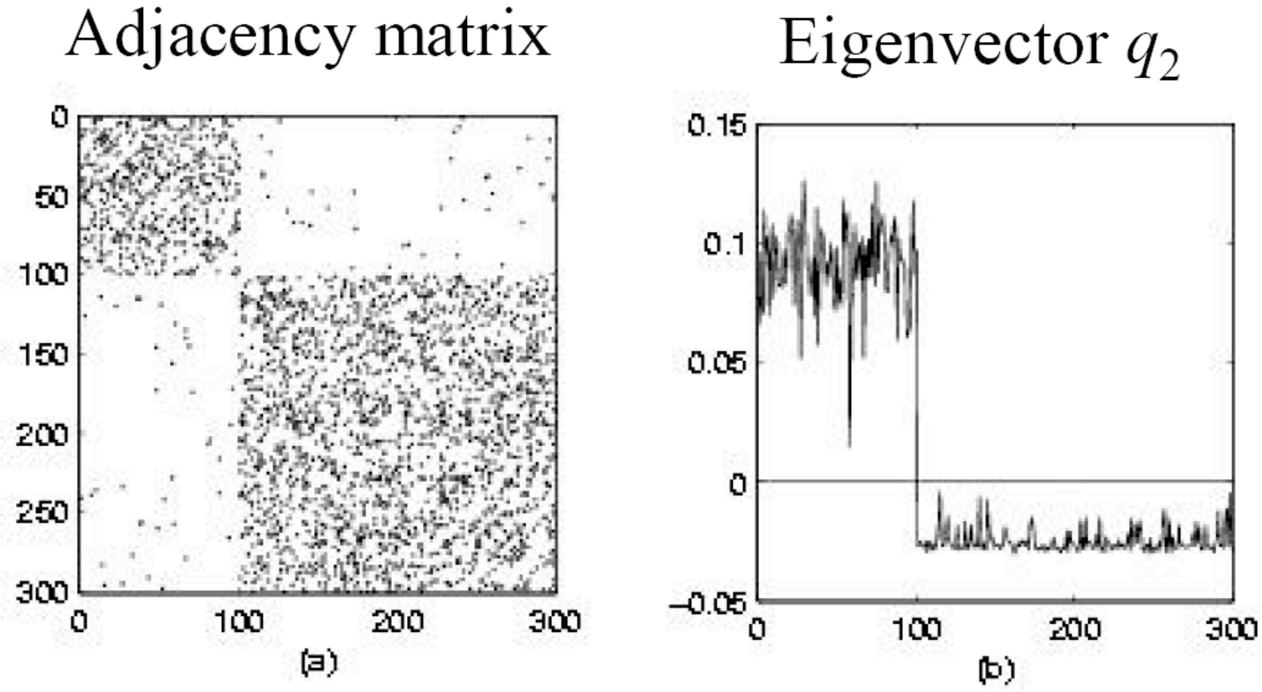
Note: only need to get an approximate eigenvector.

Actually, there is a version for any test vector:

Thm.[Mihail] Let x be such that $< x, 1 >_D = 0$. Then there is a cut along x that satisfies $\frac{x^T L_G x}{x^T D x} \geq \phi^2(S)/8$.

Spectral graph partitioning

Cluster based on the 2nd eigenvector:


Adjacency matrix

Eigenvector q_2

(a)

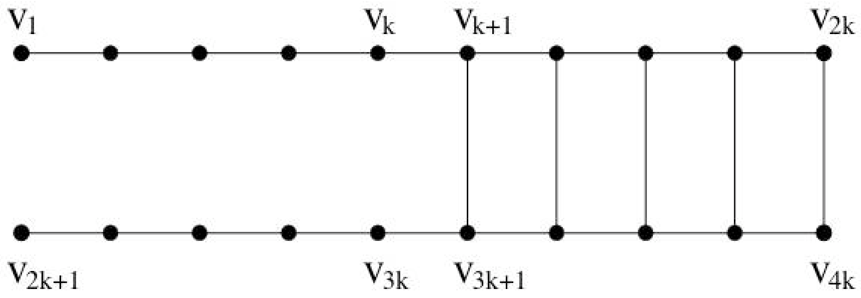
(b)

Note: “Looks” like k-means when cuts are well-balanced.

How bad can spectral be?

Guattery and Miller (1998)

- exhibit n-node graph with spectral bisection cut $O(n^{2/3})$ edges, versus optimal of $O(n^{1/3})$; takes advantage of spectral's confusion between long paths and deep cuts



Spielman and Teng (1996)

- Spectral partitioning "works" on bounded degree planar graphs and well-shaped finite element meshes, i.e., nice geometries where it was traditionally applied

An “embedding” view of spectral

Use Rayleigh quotient to characterize λ_1:

$$\lambda_1 = \min_{x \perp D1} \frac{\sum_{i \sim j} (x_i - x_j)^2}{\sum_i x_i^2 d_i}$$

Interpretation:

- Minimize “mixing” subject to variance constraint
- Embed graph on a line and cut
- But duality not tight

But since x ⊥ D1, this is equivalent to:

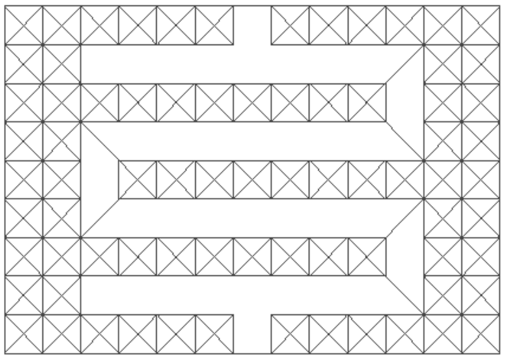
$$\frac{\lambda_1}{\mathrm{vol(G)}} = \min_{x \perp D1} \frac{\sum_{i \sim j} (x_i - x_j)^2}{\sum_{i,j} (x_i - x_j)^2 d_i d_j}$$

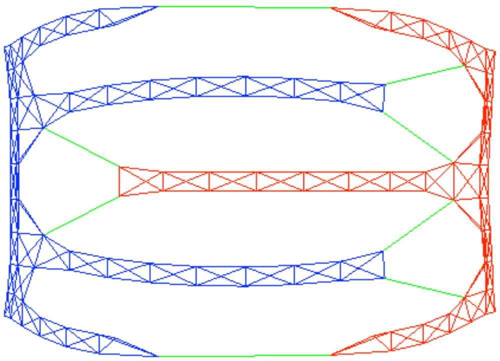
Interpretation:

- Minimize “mixing” subject to “mixing” in complete graph K_n
- Embed graph in K_n
- Duality tighter (can also see this in dual later)

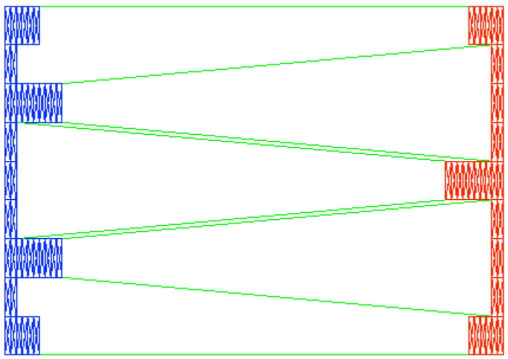
“Regularization” and spectral methods

- regularization properties: spectral embeddings stretch along directions in which the random-walk mixes slowly
  - –Resulting hyperplane cuts have "good" conductance cuts, but may not yield the optimal cuts





spectral embedding



notional flow based embedding

Local improvement methods

Kernighan and Lin (1960s) and Fiduccia and Matheyses (1970s)

- multi-pass heuristic to avoid some local minimum, but not necessarily find global optimum

Johnson et al (1990)

- Graphs up to 1000 nodes. Simulated Annealing good on random graphs, and KL work well on geometric/spacelike graphs

Lang-Rao (1993), etc.

- FM worse than flow methods on medium-sized graphs since local minimum problems lead to many small patches

1990s: Multi-resolution FM does better job of finding globally coherent solutions -> Metis

Multiresolution methods

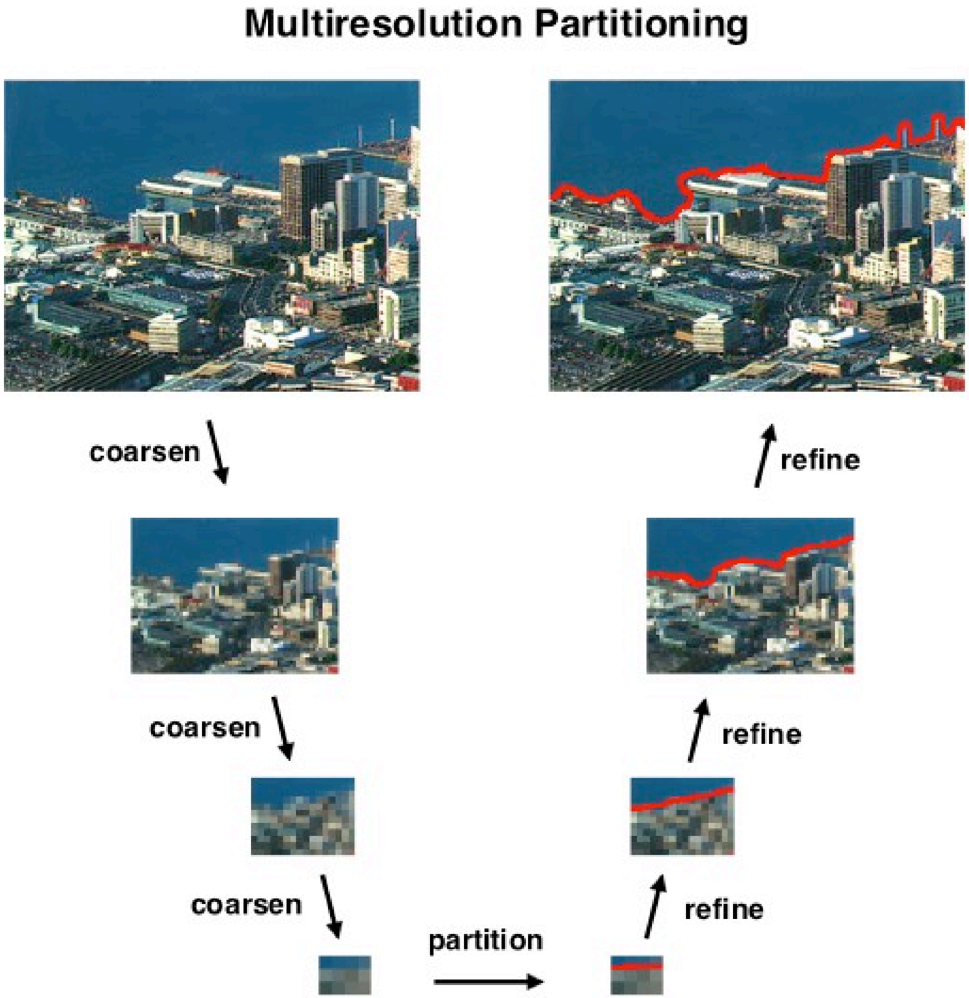
Chaco (1993)

- use multiresolution ides from Linear Algebra to couple local search with long range structure

Metis (1995)

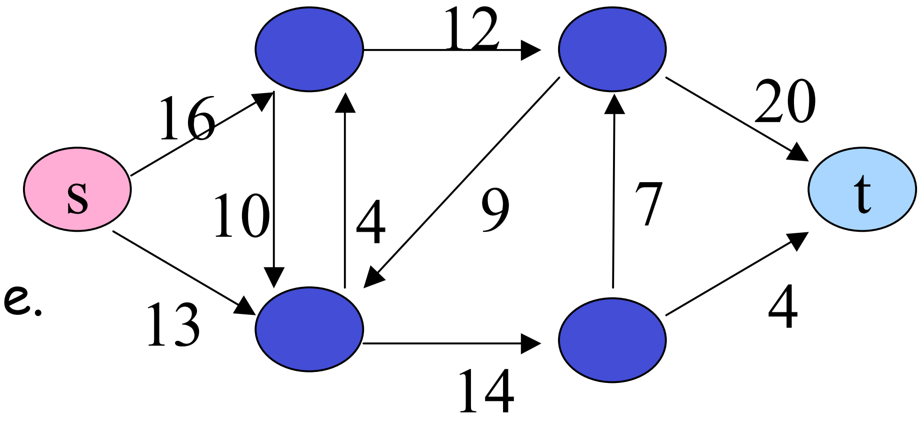
- coarsening by contracting edges (like Karger's mincut algorithm)
- very fast, and better cuts than Vanilla Spectral

Graclus, etc similar



Maximum flow problem

- Directed graph G=(V,E).
- Source s ε V, sink t ε V.
- Capacity c(e) ε $\mathbf{Z}^+$ for each edge e.
- Flow: function f: E -> **N** s.t.
- For all e: f(e) ≤ c(e)
- For all v, except s and t: flow into v = flow out of v
- Flow value: flow out of s
- Problem: find flow from s to t with maximum value



Important Variant: Multiple Sources and Multiple Sinks

Solving maximum flow problems

Single commodity flow

- Linear Programming, Ford-Fulkerson, Edmonds-Karp, Many Push-Relabel Algorithms
- MaxFlow = Min Cut

Multiple commodity flow problem

- Several different versions
- MaxFlow ≈ MinCut (up to log(k) factor for k-commodities (LR88))

Flow and graph partitioning

Single commodity flow:

- Do single commodity flow computation on all 2^n cuts and return best

Multi-commodity flow:

- Route flow between "all pairs" - n(n-1)/2 at once and then cut edges that are most congested
- log(n) gap leads to log(n) approximation guarantee
- can detect solution if bottleneck forces those edges to be more congested than average
- for expander graphs, average edge congestion is lg(n) worst than that forces by bottleneck (*so achieve worst-case guarantee*)

IP and LP view of flow

Let: x(e) = 0,1, for eεE, depending on whether edge e is cut

y(i) = 0,1, for i ε k (commodities), depending if commodity i disconnected

P_i, i ε k, is set of paths s_i to t_i

An Integer Program:

$$\begin{aligned} \min \quad & \frac{\sum_{e\in E} c(e)x(e)}{\sum_{i=1}^{k} d(i)y(i)} \\ \text{s.t.} \quad & \sum_{e\in P} x(e) \geq y(i), \forall P \in P_i \\ & y(i) \in \{0,1\}, i \in [k] \\ & x(e) \in \{0,1\}, e \in E \end{aligned}$$

A Linear Program:

$$\begin{aligned} \min \quad & \sum_{e\in E} c(e)x(e) \\ \text{s.t.} \quad & \sum_{i=1}^{k} d(i)y(i) = 1 \\ & \sum_{e\in P} x(e) \geq y(i), \forall P \in P_i \\ & y(i) \geq 0 \text{ and } x(e) \geq 0 \end{aligned}$$

An “embedding“ view of flow

Theorem: (Bourgain)

Every n-point metric space embeds into L1 with distortion O(log(n)).

Flow-based algorithm to get sparsest cuts.

(1) Solve LP to get distance d:VxV->R+.

(2) Obtain L1 embedding using Bourgain’s constructive theorem

(3) Perform an appropriate “rounding.”

Thus, it boils down to an embedding and expanders are worst.

Implementing these ideas

Spectral

- eigenvector code, e.g., Matlab, LAPACK, etc
- ≈ O(nonzeros) time to compute few eigenvectors

Metis

- nontrivial publicly-available and very usable code
- very fast in practice (tricky to analyze running time)

Flow

- Single-commodity: roughly $O(n^{3/2})$ time
- Multi-commodity: roughly $O(n^2)$ time

LPs, SDPs, etc good for theory & understanding basic ideas -- in practice, one typically depend on *high-quality numerical code.*

What is a good partitioning algorithm?

Theory says:

- Flow-based methods - since always give O(lg n) guarantee.
- Spectral methods may be ok on expanders, since quadratic of a constant is a constant

Practice says:

- Spectral methods - fast, robust, denoise, so method of choice
- Don't know or care about max-flow.

Graph partitioning highlights a deep theory-practice disconnect (and also a deep algorithmic-statistical disconnect) - they don't even qualitatively agree.

Comparison of "spectral" versus "flow"

Spectral:

- Compute an eigenvector
- "Quadratic" worst-case bounds
- Worst-case achieved -- on "long stringy" graphs
- Embeds you on a line (or complete graph)

Flow:

- Compute a LP
- O(log n) worst-case bounds
- Worst-case achieved -- on expanders
- Embeds you in L1

Two methods -- complementary strengths and weaknesses

- What we compute will be determined at least as much by as the approximation algorithm we use as by objective function.

Extensions of the basic ideas

Cut improvement algorithms

- Given an input cut, find a good one nearby or certify that none exists

Local algorithms and locally-biased objectives

- Run in a time depending on the size of the output and/or are biased toward input seed set of nodes

Combining spectral and flow

- to take advantage of their complementary strengths

Apply ideas to other objective functions

Cut-improvement algorithms

Given a graph G=(V,E) and a cut T⊂V, find a "good" conductance cut that is "near" T, or produce a certificate that none exists.

Prior work: flow-based improvement methods

- GGT89 - can find best subset S⊆T with minimum conductance in poly time
- LR04 - implement related method and show it's good at improving cuts from Metis
- AL08 - single-commodity flows to get bounds of the above form

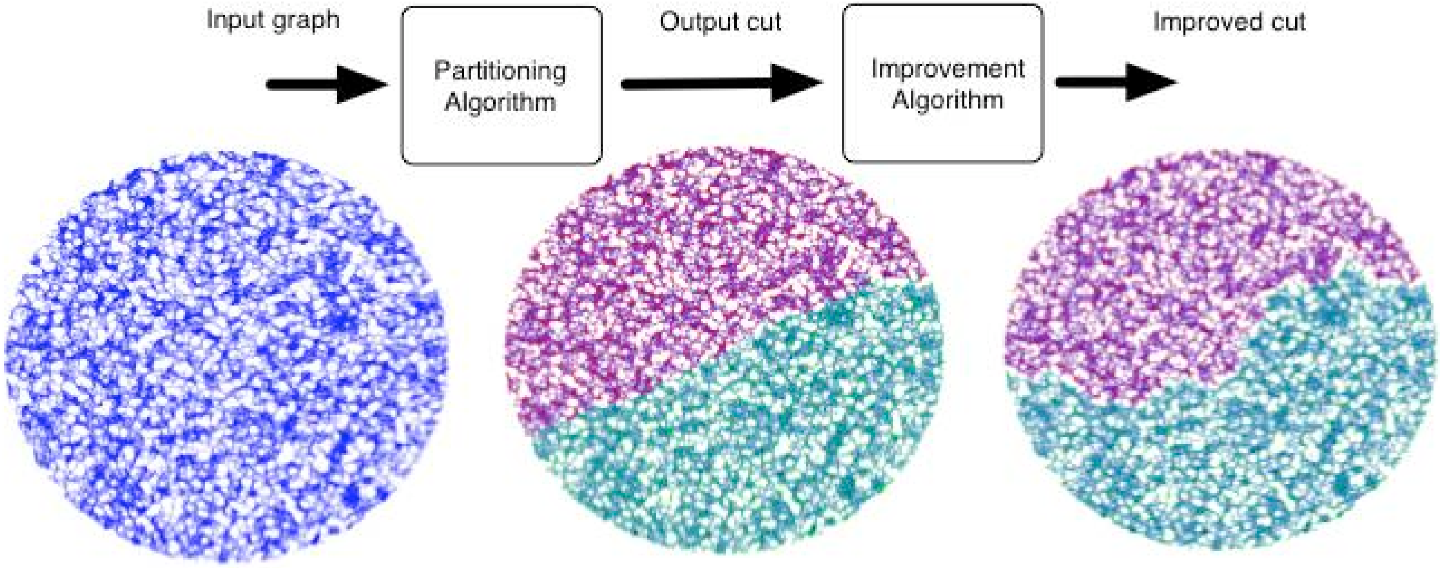
Uses of flow-based cut-improvement algorithms

- algorithmic primitive in fast versions of theoretically best partitioning algorithms
- identifying community structure in large social and information networks

Flow "improvement" algorithms

Andersen and Lang (2008)

- Modified quotient cost - cost relative to input set A penalizes sets for including vertices outside of A
- Constructing and solving sequence of s-t min cut problems in augmented graph



Flow "improvement" algorithms

Andersen and Lang (2008)

- Modified quotient cost - cost relative to input set A penalizes sets for including vertices outside of A

- Constructing and solving sequence of s-t min cut problems in augmented graph

Theorem: Let C be any set whose intersection with the proposed set A s.t.

$$\frac{\pi(A \cap C)}{\pi(C)} \geq \frac{\pi(A)}{\pi(V)} + \epsilon$$

Then, the set S returned has quotient cost almost as small as C:

$$Q(S) \leq \frac{1}{\epsilon} Q(C)$$

Local clustering algorithms

Spielman and Teng (2008)

- local algorithm finds a solution containing or near a given vertex without looking at the entire graph
- running time is “nearly linear” in the size of output cluster
- gets Cheeger-like quadratically-good approximation guarantees
- Based on Lovasz-Simonovitz (90,93) random walk

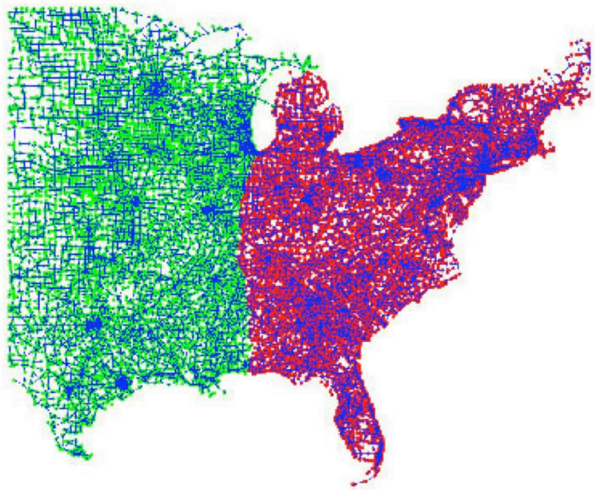
Local spectral methods

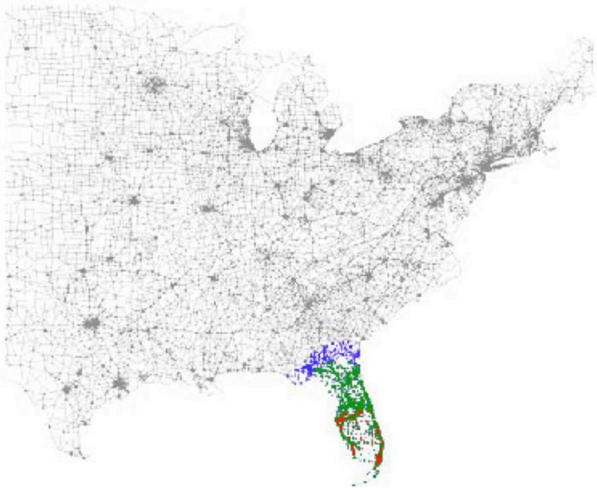
Local spectral methods - provably-good local version of global spectral

ST04: truncated "local" random walks to compute locally-biased cut

ACL06: approximate locally-biased PageRank vector computations

Chung08: approximate heat-kernel computation to get a vector





Spectral “improvement” algorithms and optimization programs

Global Spectral and Flow

- Can write objective function and optimization
- Algorithm solves that objective function

Local and Improvement Methods

- More “operationally” defined using steps similar to global but subject to constraints (locality constraints of modified objective

Can we write these as optimization programs?

Recall spectral graph partitioning

The basic optimization problem:

$$\begin{aligned} \text{minimize} \quad & x^T L_G x \\ \text{s.t.} \quad & \langle x, x \rangle_D = 1 \\ & \langle x, 1 \rangle_D = 0 \end{aligned}$$

- Relaxation of:

$$\phi(G) = \min_{S \subset V} \frac{E(S, \bar{S})}{Vol(S)Vol(\bar{S})}$$

- Solvable via the eigenvalue problem: $\mathcal{L}_G y = \lambda_2(G) y$

- Sweep cut of second eigenvector yields:

$$\lambda_2(G)/2 \leq \phi(G) \leq \sqrt{8\lambda_2(G)}$$

Also recall Mihail's sweep cut for a general test vector:

Thm.[Mihail] Let x be such that $< x, 1 >_D = 0$. Then there is a cut along x that satisfies $\frac{x^T L_G x}{x^T D x} \geq \phi^2(S)/8$.

Geometric correlation and generalized PageRank vectors

Given a cut T, define the vector:

$$s_T := \sqrt{\frac{\mathrm{vol}(T)\mathrm{vol}(\bar{T})}{2m}} \left(\frac{1_T}{\mathrm{vol}(T)} - \frac{1_{\bar{T}}}{\mathrm{vol}(\bar{T})} \right)$$

Can use this to define a geometric notion of correlation between cuts:

$$< s_T, 1 >_D = 0$$

$$< s_T, s_T >_D = 1$$

$$< s_T, s_U >_D = K(T,U)$$

Defn. Given a graph $G = (V,E)$, a number $\alpha \in (-\infty, \lambda_2(G))$ and any vector $s \in R^n$, $s \perp_D 1$, a ***Generalized Personalized PageRank (GPPR)*** vector is any vector of the form

$$p_{\alpha,s} := (L_G - \alpha L_{K_n})^+ Ds.$$

- PageRank: a spectral ranking method (regularized version of second eigenvector of L_G)
- Personalized: s is nonuniform; & generalized: teleportation parameter α can be negative.

Local spectral partitioning *ansatz*

Mahoney, Orecchia, and Vishnoi (2010)

Primal program:

$$\begin{aligned} \text{minimize} \quad & x^T L_G x \\ \text{s.t.} \quad & < x, x >_D = 1 \\ & < x, s >_D^2 \geq \kappa \end{aligned}$$

Dual program:

$$\begin{aligned} \max \quad & \alpha - \beta(1-\kappa) \\ \text{s.t.} \quad & L_G \succeq \alpha L_{K_n} - \beta \left(\frac{L_{K_T}}{\mathrm{vol}(\bar{T})} + \frac{L_{K_{\bar{T}}}}{\mathrm{vol}(T)} \right) \\ & \beta \geq 0 \end{aligned}$$

Interpretation:

- Find a cut well-correlated with the seed vector s.

- If s is a single node, this relax:

$$\min_{S \subset V, s \in S, |S| \leq 1/k} \frac{E(S,\bar{S})}{Vol(S)Vol(\bar{S})}$$

Interpretation:

- Embedding a combination of scaled complete graph K_n and complete graphs T and $\underline{T}$ (K_T and $K_{\underline{T}}$) - where the latter encourage cuts near (T,$\underline{T}$).

Main results (1 of 2)

Mahoney, Orecchia, and Vishnoi (2010)

Theorem: If x^* is an optimal solution to LocalSpectral, it is a GPPR vector for parameter α, and it can be computed as the solution to a set of linear equations.

Proof:

(1) Relax non-convex problem to convex SDP

(2) Strong duality holds for this SDP

(3) Solution to SDP is rank one (from comp. slack.)

(4) Rank one solution is GPPR vector.

Main results (2 of 2)

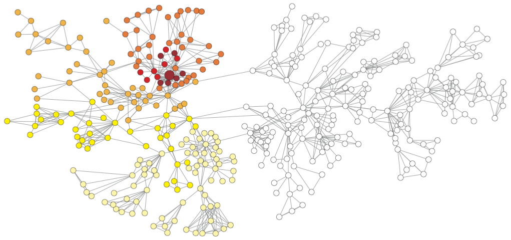
Mahoney, Orecchia, and Vishnoi (2010)

Theorem: If x^* is optimal solution to LocalSpect(G,s,κ), one can find a cut of conductance $\leq 8\lambda(G,s,\kappa)$ in time $O(n \lg n)$ with sweep cut of x^*.

Upper bound, as usual from sweep cut & Cheeger.

Theorem: Let s be seed vector and κ correlation parameter. For all sets of nodes T s.t. $\kappa' := \langle s, s_T \rangle_D^2$, we have: $\phi(T) \geq \lambda(G,s,\kappa)$ if $\kappa \leq \kappa'$, and $\phi(T) \geq (\kappa'/\kappa)\lambda(G,s,\kappa)$ if $\kappa' \leq \kappa$.

Lower bound: Spectral version of flow-improvement algs.



Other “Local” Spectral and Flow and “Improvement” Methods

Local spectral methods - provably-good local version of global spectral

ST04: truncated“local” random walks to compute locally-biased cut

ACL06/Chung08 : locally-biased PageRank vector/heat-kernel vector

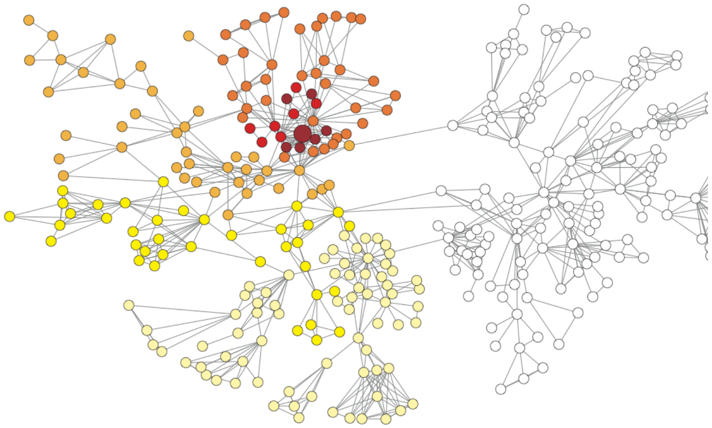
Flow improvement methods - *Given a graph G* and a partition, find a “nearby” cut that is of similar quality:

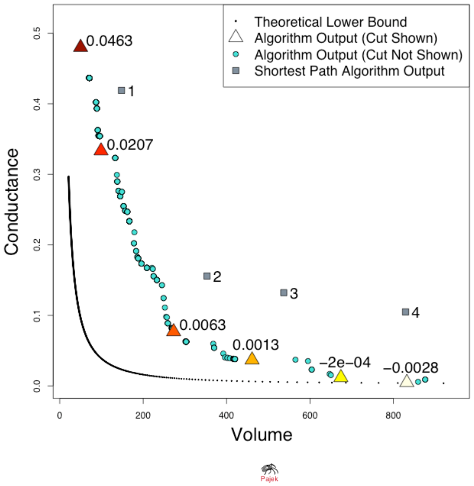
GGT89: find min conductance subset of a “small” partition

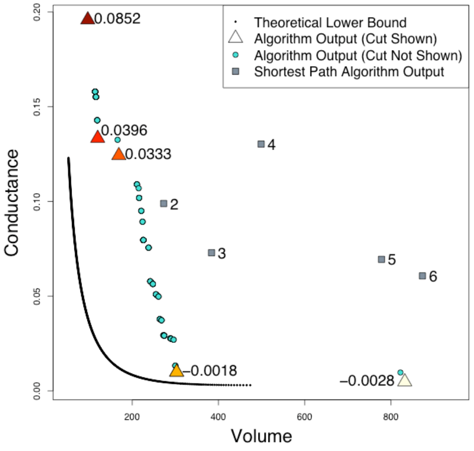
LR04,AL08: find “good” “nearby” cuts using *flow-based methods*

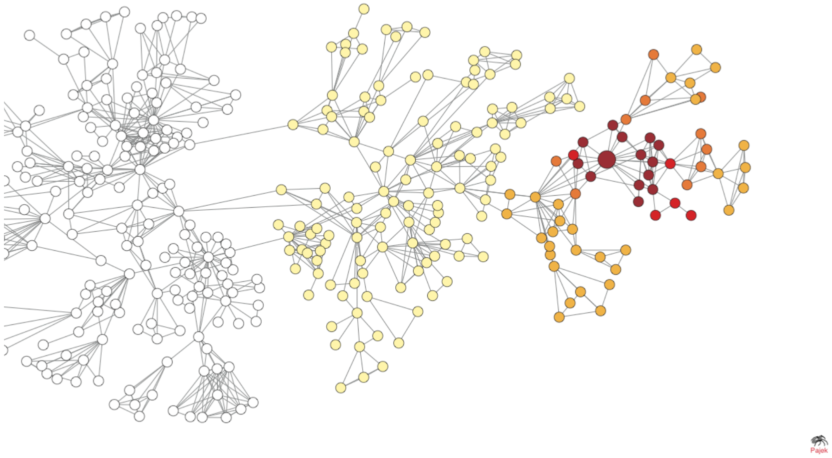
Optimization ansatz ties these two together (but is *not* strongly local in the sense that computations depend on the size of the output).

Illustration on small graphs









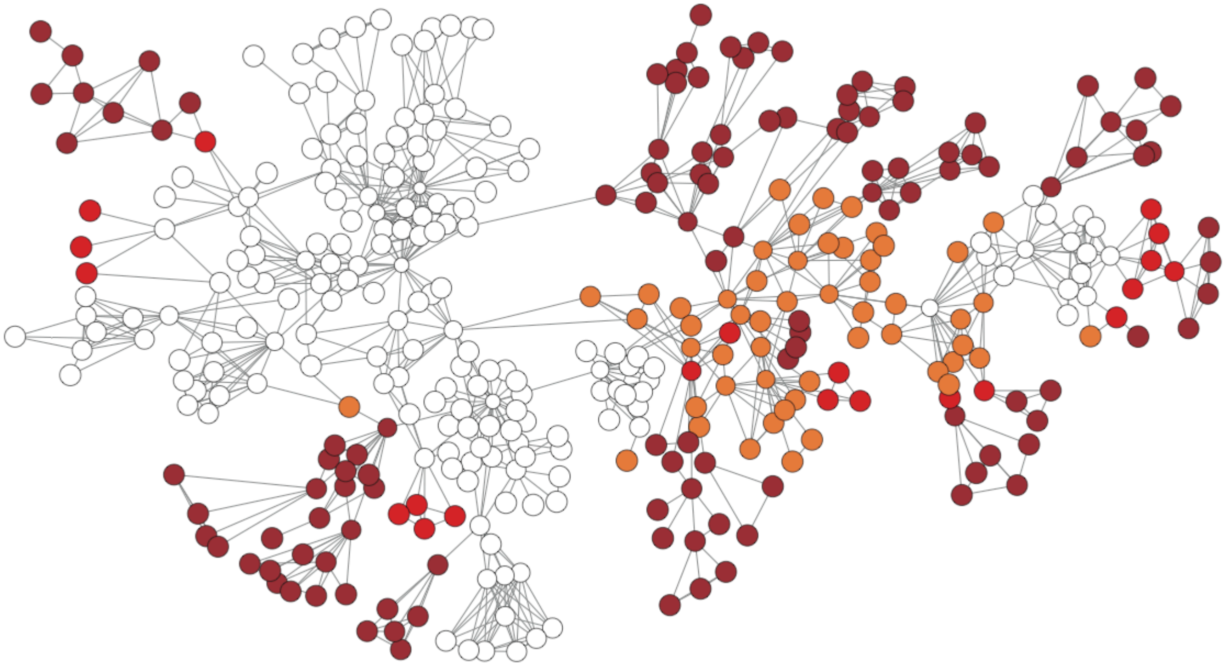
- Similar results if we do local random walks, truncated PageRank, and heat kernel diffusions.

- Often, it finds "worse" quality but "nicer" partitions than flow-improve methods. (Tradeoff we'll see later.)

Illustration with general seeds

- Seed vector doesn't need to correspond to cuts.

- It could be any vector on the nodes, e.g., can find a cut "near" low-degree vertices with $s_i = -(d_i-d_{av})$, $i\varepsilon[n]$.



Comparison with Flow-Improve

AL08 (implicitly) measure how much more of C in T than expected:

Given two cuts $(C, \bar{C})$ and $(T, \bar{T})$ s.t. $\mathrm{vol}(C) \le \mathrm{vol}(\bar{C})$ and $\mathrm{vol}(T) \le \mathrm{vol}(\bar{T})$:

$$F(C,T) := \frac{\mathrm{vol}(T)}{\mathrm{vol}(C)} \left(\frac{\mathrm{vol}(C \cap T)}{\mathrm{vol}(T)} - \frac{\mathrm{vol}(C \cap \bar{T})}{\mathrm{vol}(\bar{T})} \right).$$

Spectral and flow correlation measures are related:

$$\textbf{Lemma:}\ \frac{\mathrm{vol}(T)}{\mathrm{vol}(C)} K(C,T) \le F(C,T)^2 \le \frac{2\mathrm{vol}(T)}{\mathrm{vol}(C)} K(C,T)$$

Notes (aside from that this is eigenvector computation):

- Spectral better (in theory) if $\phi(C)$ large, e.g., G an expander
- Spectral better if input cut volume << volume of cut we bound

Comparison with local spectral algorithms

Optimization ansatz

- is local in the sense that seed vector is local
- is *not* local in sense that computations depend on the size of output

PageRank, HeatKernel, Truncated Random Walks - can all be viewed as regularized versions of computing second eigenvector (see below)

Previous algorithms introduce structured approximations to approximate PageRank, HeatKernel, Diffusions

- Question: Can these be formalized as optimization problems?

Combining spectral and flow

Arora, Rao, Vazirani (2004)

- Can we improve O(log(n)) from L1 embedding?
- Relax to L2 - No. (Not convex, so can't optimize efficiently.)
- Relax to L2^2, space of squared L2 metrics - No. (Can optimize, but "gap" is O(n). Note: not even a metric, since triangle inequality violated, but "average" squared distance is small.)

Relax to Metrics∩L2^2 - Yes!!

- Can write as SDP.
- Get O(sqrt(log(n))) approximation with a O(n$^{4.5}$) algorithm

Combining spectral and flow, cont.

Arora, Hazan, and Kale (AHK, 2004)

• multi-commodity flow implementation of expander flow framework to achieve an O(sqrt(log n)) approximation in roughly O(n^2) time

Arora and Kale (AK, 2007)

•similar ideas to give an O(log n) approximation more generally

Khandekar, Rao, and Vazirani (KRV, 2006)

•polylogarithmic single commodity max-flow computations iteratively to embed an expander flow, O($\log^2$ n) approximation in roughly O($n^{3/2}$) time.

Orecchia, Schulman, Vazirani, and Vishnoi (OSVV, 2008)

• related algorithm also performs only polylogarithmic single commodity max-flow computations to achieve an O(log n) approximation.

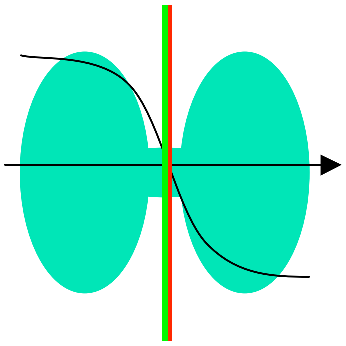
OSVV "spectral-flow" partitioning

Orecchia, Schulman, Vazirani, and Vishnoi (2008) - variant of Arora, Rao, Vazirani (2004); also Lang, Mahoney, Orecchis (2009)

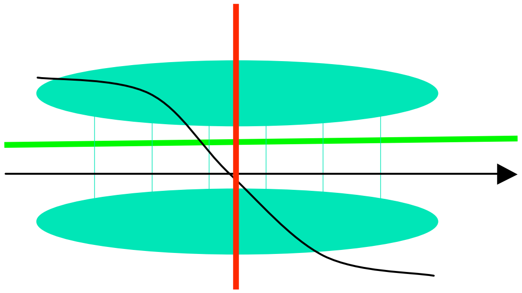
SPECTRAL

- 2nd eigenvector →
- Spectral cut 🟥
- Optimal cut 🟩

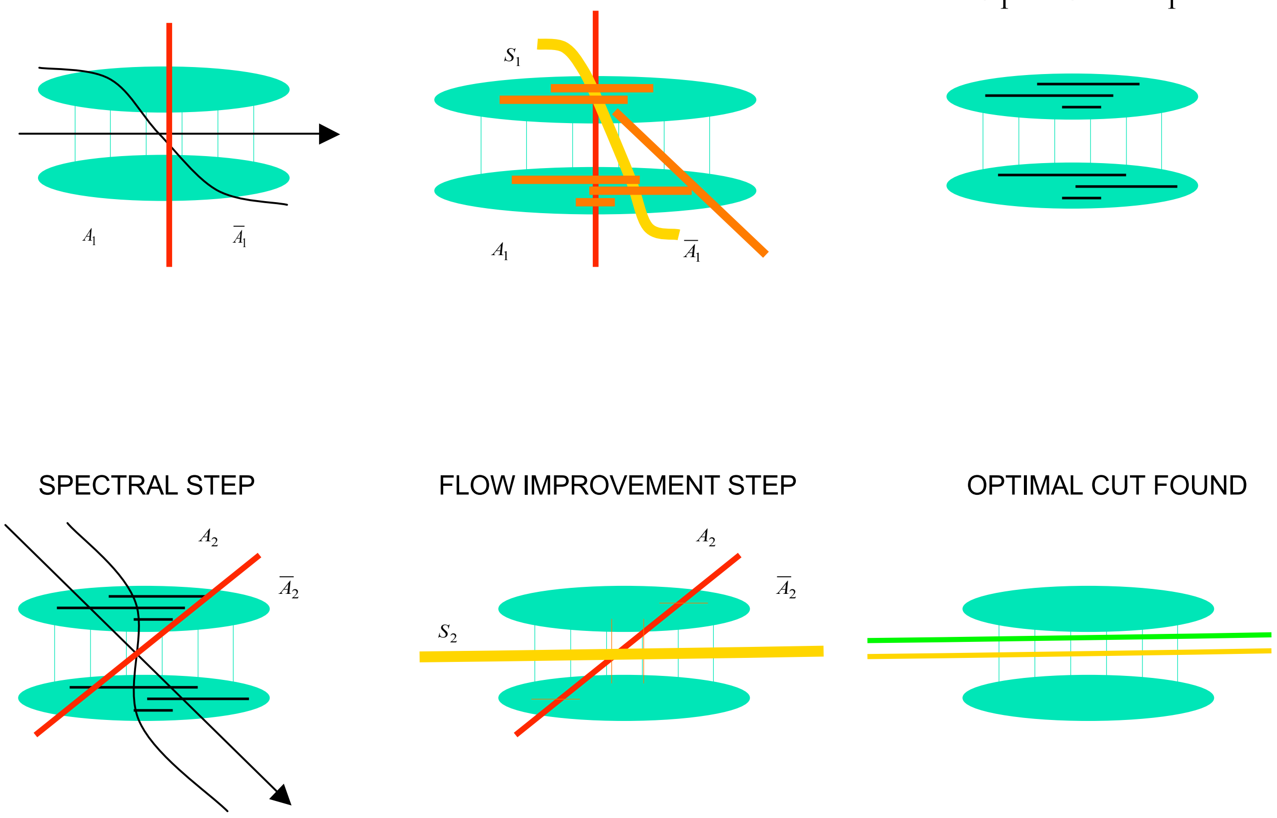
GOOD CASE



BAD CASE: LONG PATHS



OSVV



SPECTRAL STEP

FLOW IMPROVEMENT STEP

$G_1 = G + M_1$

SPECTRAL STEP

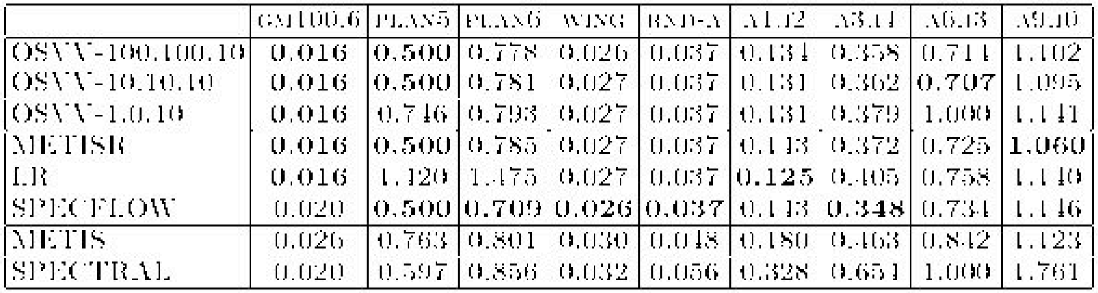
FLOW IMPROVEMENT STEP

OPTIMAL CUT FOUND

Initial evaluation of OSVV

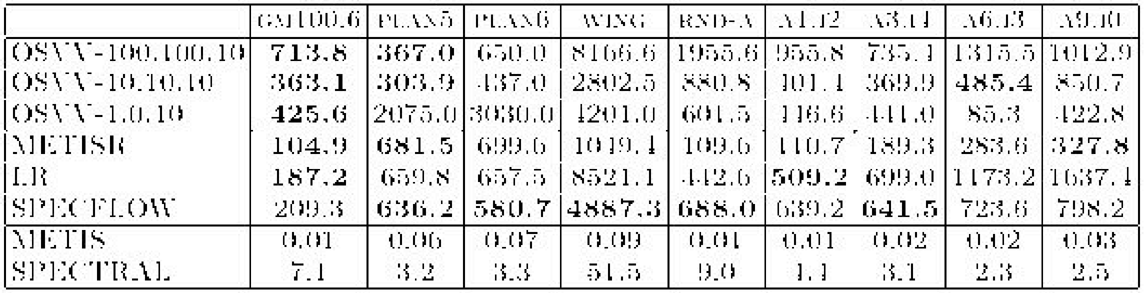
Classes of Graphs:

- GM (Guattery-Miller) graph where eigenvector methods fail.
- PLAN - Expanders with planted bisections - where LR is known to fail
- WING - finite element mesh
- RND - Random Geometric Graph
- Random geometric graph with random edges added



	GM100.6	PLAN5	PLAN6	WING	RND-A	A1.12	A3.14	A6.13	A9.10
OSVV-100.100.10	**0.016**	**0.500**	0.778	0.026	0.037	0.134	0.358	0.714	1.102
OSVV-10.10.10	**0.016**	**0.500**	0.781	0.027	0.037	0.131	0.362	**0.707**	1.095
OSVV-1.0.10	**0.016**	0.746	0.793	0.027	0.037	0.131	0.379	1.000	1.141
METISR	**0.016**	**0.500**	0.785	0.027	0.037	0.113	0.372	0.725	**1.060**
LR	**0.016**	1.120	1.475	0.027	0.037	**0.125**	0.405	0.758	1.140
SPECFLOW	0.020	**0.500**	**0.709**	**0.026**	**0.037**	0.113	**0.348**	0.734	1.146
METIS	0.026	0.763	0.801	0.030	0.048	0.180	0.463	0.842	1.123
SPECTRAL	0.020	0.597	0.856	0.032	0.056	0.328	0.654	1.000	1.761

Fig. 2. The best score found by multiple tries (see caption of Figure 3) of each algorithm. First and 2nd-place for each graph are highlighted in red and blue respectively. Scores are given to 3 decimal digits. OSVVparameters are described as OSVV$-\eta.init.s$



	GM100.6	PLAN5	PLAN6	WING	RND-A	A1.12	A3.14	A6.13	A9.10
OSVV-100.100.10	**713.8**	**367.0**	650.0	8166.6	1955.6	955.8	735.4	1315.5	1012.9
OSVV-10.10.10	**363.1**	303.9	437.0	2802.5	880.8	101.1	369.9	**485.4**	850.7
OSVV-1.0.10	**425.6**	2075.0	3030.0	4201.0	601.5	116.6	441.0	85.3	422.8
METISR	**104.9**	**681.5**	699.6	1019.4	109.6	110.7	189.3	283.6	**327.8**
LR	**187.2**	659.8	657.5	8521.1	442.6	**509.2**	699.0	1173.2	1637.4
SPECFLOW	209.3	**636.2**	**580.7**	**4887.3**	**688.0**	639.2	**641.5**	723.6	798.2
METIS	0.01	0.06	0.07	0.09	0.01	0.01	0.02	0.02	0.03
SPECTRAL	7.1	3.2	3.3	51.5	9.0	1.4	3.1	2.3	2.5

Fig. 3. Total run time in seconds for OSVV $-\eta.init.s$ (10 tries). METISR (10000 tries). LR (10 tries). SPECFLOW (Eigensolver $-$ 1000 flow roundings). METIS (1 try). SPECTRAL (Eigensolver + 3 sweep roundings).

Connections with boosting

Iterative nature of "fast ARV" algorithms can be done with cut-matching game

- Cut player - choose bisection (to make game last long)
- Matching player - choose matching to add to G, i.e., G'=G+M
- Game stop when G' is an expander

Connections b/w game theory, online learning, & boosting

- Freund and Schapire (1996), Warmuth et al (2008)

Online algorithms: practice follows theory quite closely

- **Question**: can this be used as a model to understand statistical properties implicit in approximation algorithms more generally?

Other applications of spectral and flow

Recall: graph partitioning was a “hydrogen atom”

- For studying spectral/flow/etc relaxations to combinatorial problems
- Much of this “spectral” and “flow” structure inherited by approximations to other optimization problem

Spectral: NCut, k-means, Transductive Learning, Modularity relaxations, (esp, in ML), etc.

Flow: Lots of graph approximation algorithms, (in TCS)

Another application of similar ideas: Finding *dense* sub-graphs

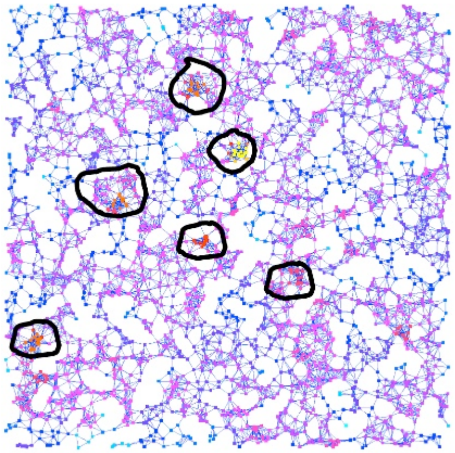
Andersen and Chellapilla (2009), Andersen (2008), Charikar (2000), Kannan and Vinay (1999), GGR (1998), Goldberg (1984), etc.

Definition: Given $G = (V, E)$, an undirected graph, define the *density* $f(S)$ of $S \subset V$ to be

$$f(S) = \frac{|E(S, \bar{S})|}{|S|}.$$

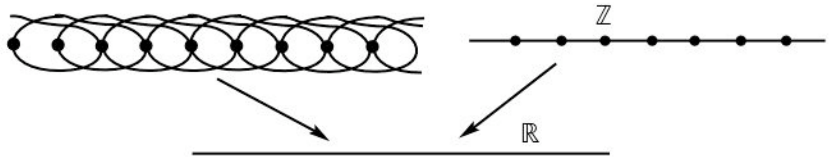
Given $G = (V, E)$, a directed bipartite graph, define the *density* $d(S, T)$ of induced subgraph (S, T) to be

$$d(S, T) = \frac{|E(S, T)|}{\sqrt{|S|}\sqrt{|T|}}.$$



- Optimize f(S) with max-flow or parametric **flow**.
- Greedy approx algorithms optimize f(S) and d(S,T).
- Global/Local **spectral** algs approximate d(S,T) - more amenable to spectral algorithms.

Also, tradeoff dense versus isolated sub-graphs. (Lang and Andersen 2007).



What is the *shape* of a graph?

Can we generalize the following intuition to general graphs:

- A 2D grid or well-shaped mesh "looks like" a 2D plane*

- A random geometric graph "looks like" a 2D plane

- An expander "looks like" a clique or complete graph or a point.

The basic idea:

- If a graph embeds well in another metric space, then it "looks like" that metric space**!

*A "planar graph" is typically a very different combinatorial thing.

**Gromov (1987); Linial, London, & Rabinovich (1985); ISOMAP, LLE, LE, … (2001)

What is the *shape* of a space?

A long history:

- Euclid (BC): R^n lengths, angles, dot products, etc come from his Fifth Parallel Lines Postulate

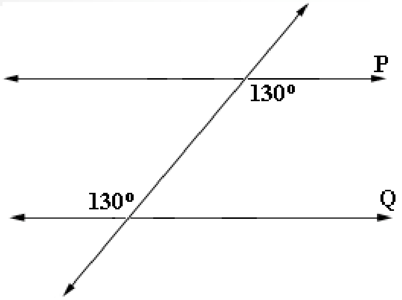
- Bolyai, Lobachevsky etc. (1830s): formulate consistent geometries with other fifth postulates

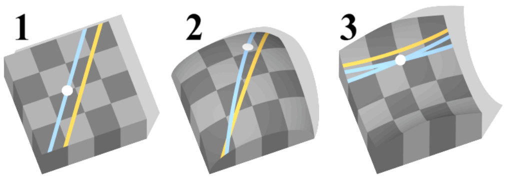
- Riemann (1850s): work on manifolds and curvature more generally

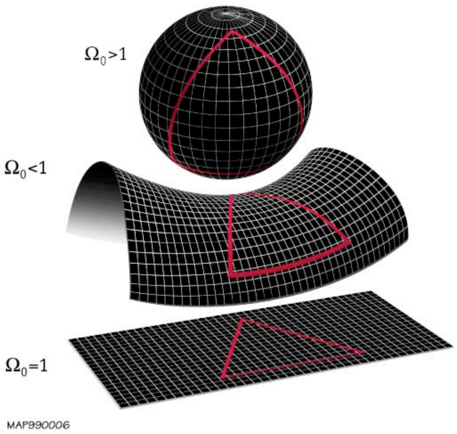
- Einstein (1910s): applications to curvature properties of physical spacetime

- Gromov (1980s): *discrete* curvature and hyperbolicity

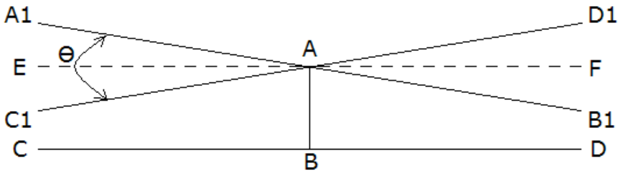
- 1990s and 2000s: applications of network curvature in routing, visualization, embedding, etc.







Hyperbolic Spaces

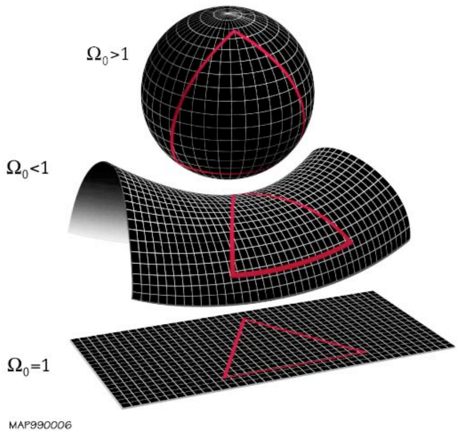


Lobachevsky and Bolyai constructed hyperbolic space - (between a point and a line, there are many “parallel” lines) - Euclid’s fifth postulate is independent of the others!

A d-dimensional metric space which is homogeneous and isotropic (looks the same at every point and in every direction) is locally identical to one of:

- Sphere
- Hyperbolic space
- Euclidean plane

} The 3 maximally symmetric geometries

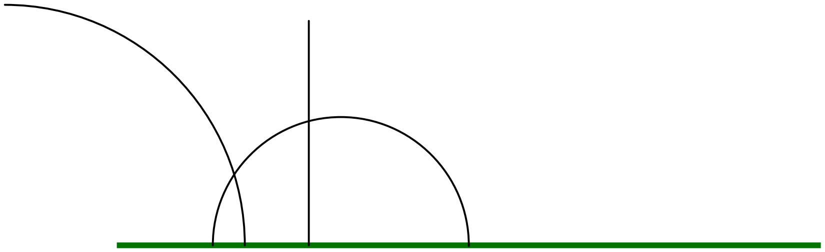


Models of the Hyperbolic Plane

UPPER HALF PLANE MODEL

- Points are {z:Im(z)>0}
- Length of a path z(t) is

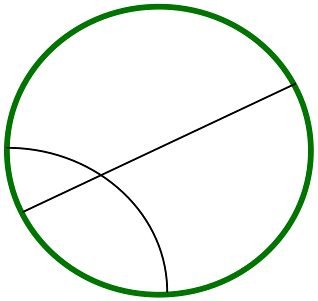
$$\int_0^1 \frac{1}{\mathrm{Im}(z)} \left\| \frac{dz}{dt} \right\| dt$$



POINCARE DISK MODEL

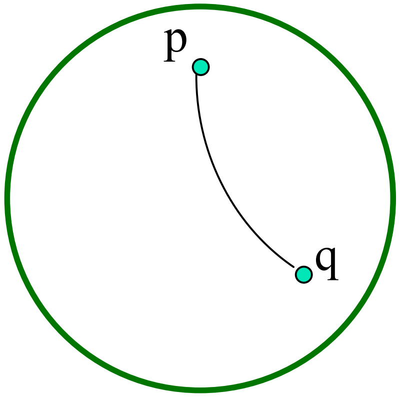
- Points are {z: |z|<1}.
- Length of a path z(t) is

$$\int_0^1 \frac{1}{1-\|z\|^2} \left\| \frac{dz}{dt} \right\| dt$$

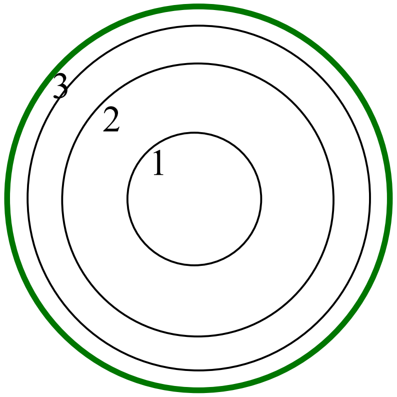


Distances in hyperbolic space

- Vectors are *longer* near the boundary.
- Shortest path from *p* to *q* bends toward the center, where vectors are shorter.
- Geodesics are circular arcs meeting the boundary at right angles.

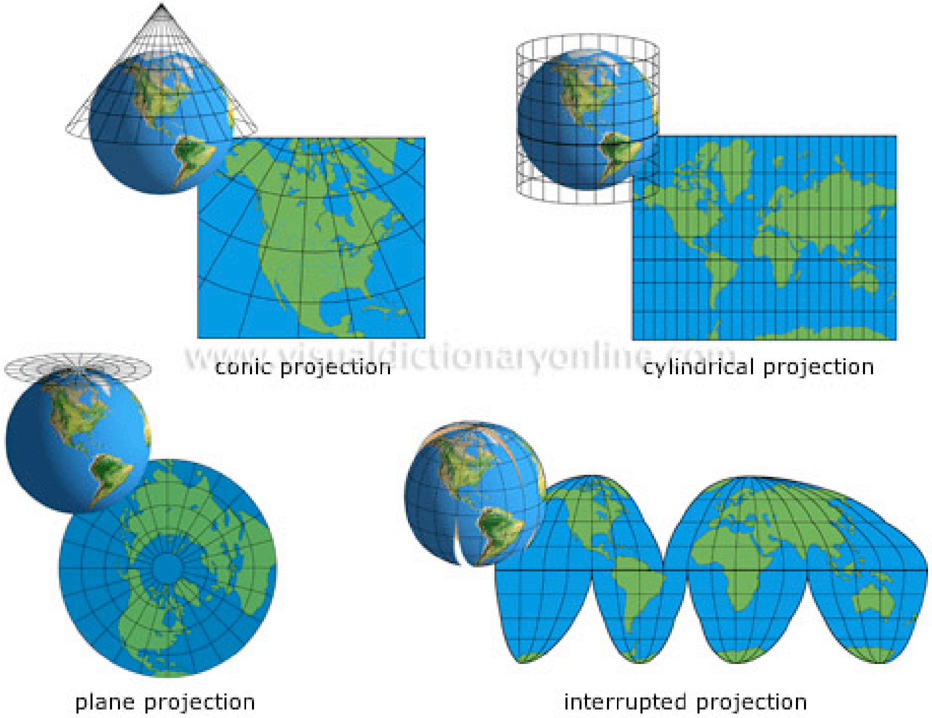


- If you draw circles of hyperbolic radius 1,2,3,… around the center of the Poincare disk, each is ≈ *e* times closer to the boundary than the previous one. Their circumferences grow exponentially!

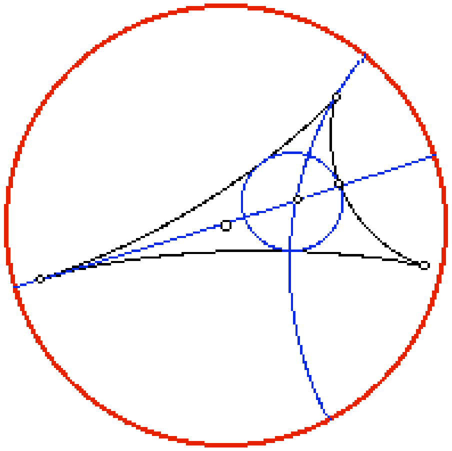


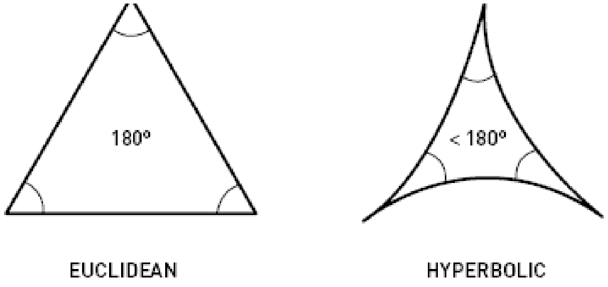
Interpreting visualizations …

Positive curvature:



Negative curvature:

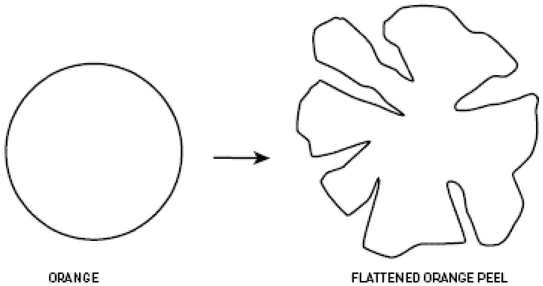




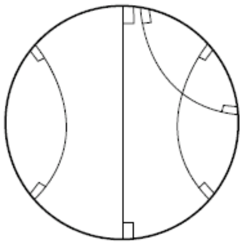
How much space is there in a space?

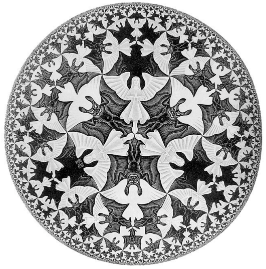
Intuitively,

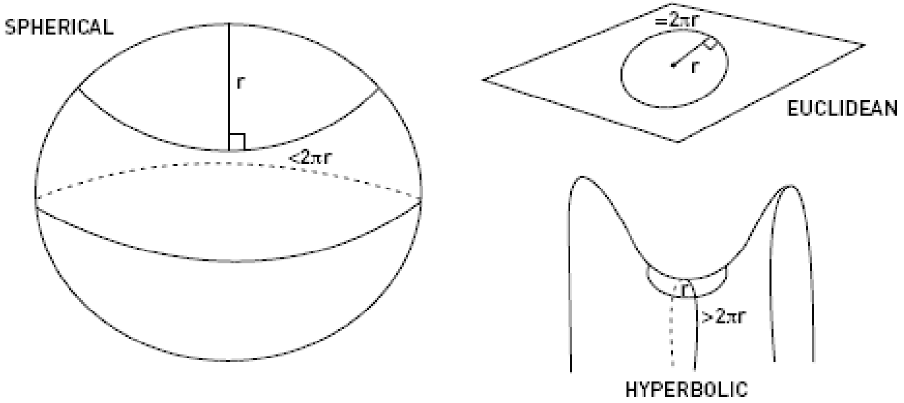
- positively-curved spaces have less space than flat spaces.



- flat spaces have less space than negatively-spaces.

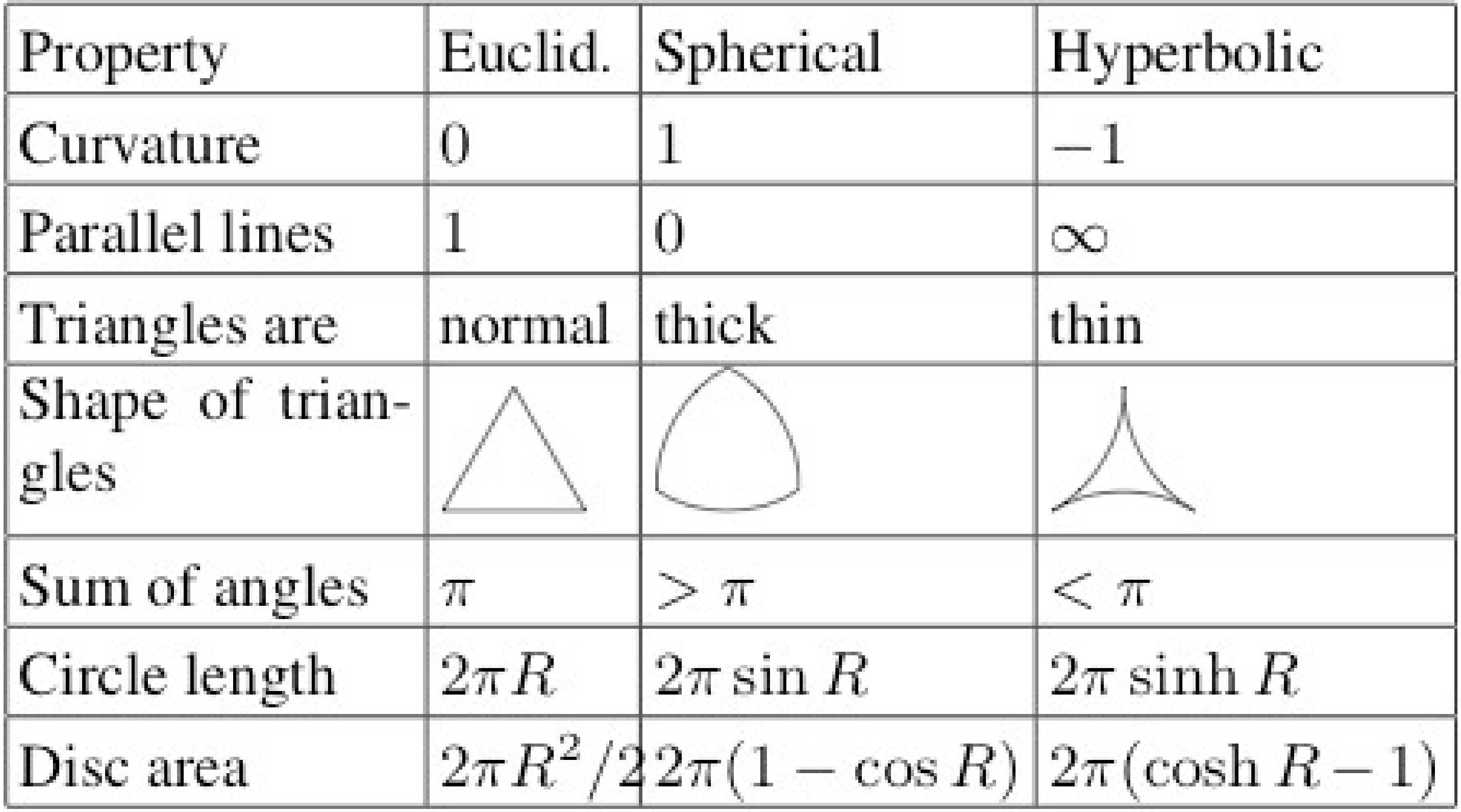






Imagine starting with a flat piece of paper and trying "cover" a sphere (you'll need to crumple it) or a saddle (you'll need to cut it to make room).

Comparison between different curvatures



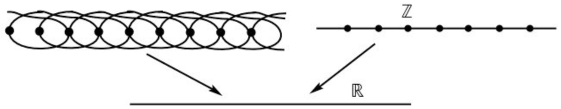
Property	Euclid.	Spherical	Hyperbolic
Curvature	0	1	-1
Parallel lines	1	0	∞
Triangles are	normal	thick	thin
Shape of triangles			
Sum of angles	π	$> \pi$	$< \pi$
Circle length	$2\pi R$	$2\pi \sin R$	$2\pi \sinh R$
Disc area	$2\pi R^2/2$	$2\pi(1 - \cos R)$	$2\pi(\cosh R - 1)$

Discrete vs. continuous

See: “Discrete Geometric Analysis,” T. Sunada (2007)

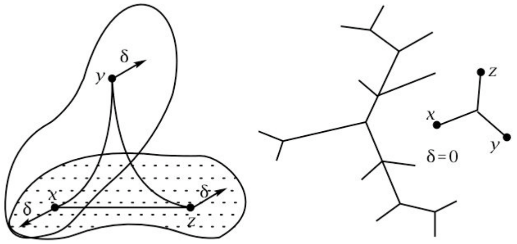
“Squint” at data with “coarse embedding”

- Line graph is “like” a line (random geometric graph is like underlying geometry).



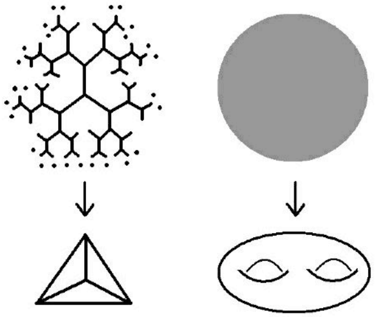
- Expander is “like” a complete graph. (Hard to visualize.)

- Hyperbolic metric is “like” tree!



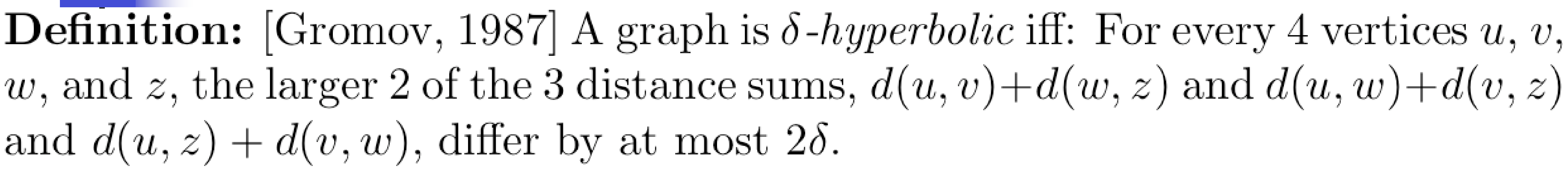
A striking example of analogy

Regular tree and Poincare disc



Graph Theory	Geometry
a regular tree X	the unit disc D with the Poincaré metric
automorphism group of X	isometry group of H
a finite regular graph	a closed Riemann surface with constant negative curvature
discrete Laplacian on X	Laplacian Δ on D
paths without backtracking	geodesics
spherical functions on X	spherical functions on H
Ihara’s zeta function for a finite regular graph	Selberg’s zeta function for a closed Riemann surface

δ-hyperbolic metric spaces



Definition: [Gromov, 1987] A graph is δ-*hyperbolic* iff: For every 4 vertices u, v, w, and z, the larger 2 of the 3 distance sums, $d(u,v)+d(w,z)$ and $d(u,w)+d(v,z)$ and $d(u,z)+d(v,w)$, differ by at most 2δ.

Things to note about δ-hyperbolicity:

- Graph property that is both *local* (by four points) and *global* (by the distance) in the graph

- Polynomial time computable - naively in $O(n^4)$ time

- Metric space embeds into a tree iff $\delta = 0$.

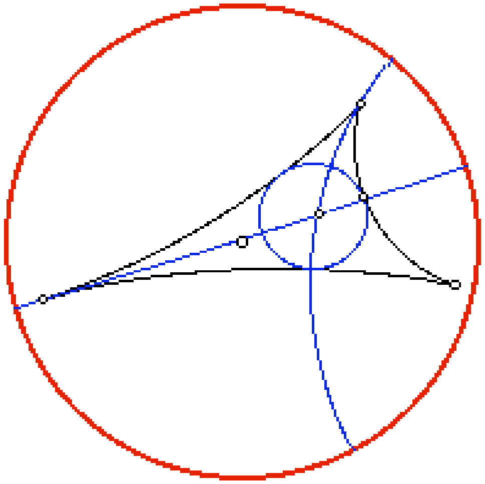
- Poincare half space in R^k is δ-hyperbolic with $\delta = \log_2 3$

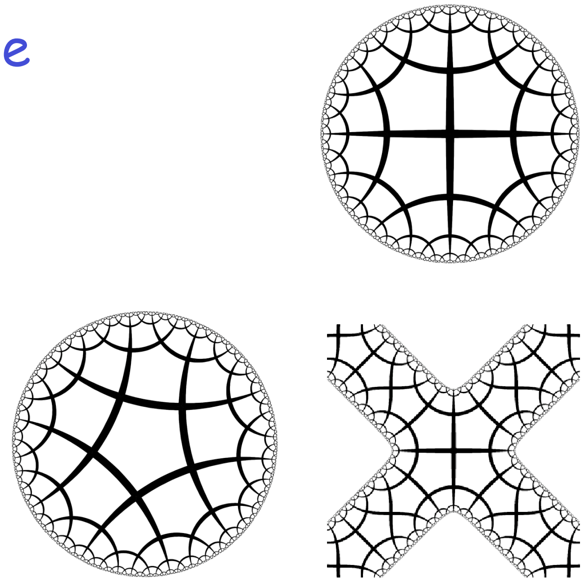
- Theory of δ-hyperbolic spaces generalize theory of Riemannian manifold with negative sectional curvature to metric spaces

δ-hyperbolic metric spaces, cont.

Theory of δ-hyperbolic spaces generalize theory of Riemannian manifold with negative sectional curvature to metric spaces.

- Measures *deviation from tree-ness* of a discrete space
- Equivalent definition in terms of δ-thin triangle condition:





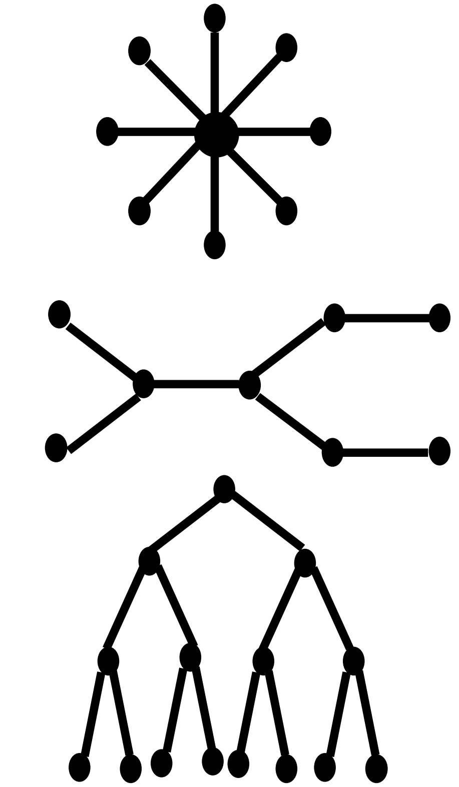
Expanders and hyperbolicity

Different concepts that really are different (Benjamini 1998) :

- Constant-degree expanders - like sparsified complete graphs
- Hyperbolic metric space - like a tree-like graph

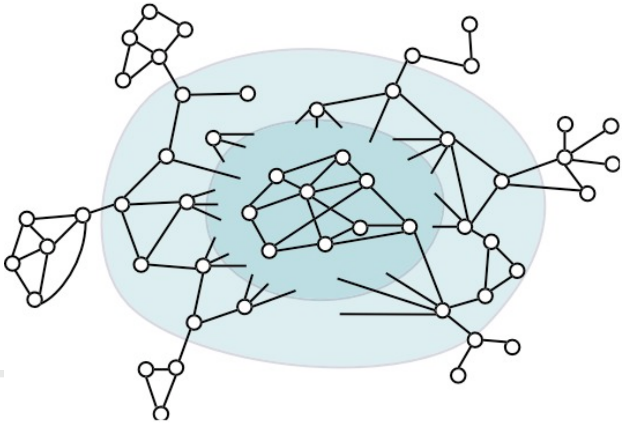
But, *degree heterogeneity enhances hyperbolicity** (so real networks will often have both properties).

Trees come in all sizes and shapes:



*Question: Does anyone know a reference that makes these connections precise?

Overview



Popular algorithmic tools with a geometric flavor

- PCA, SVD; interpretations, kernel-based extensions; algorithmic and statistical issues; and limitations

Graph algorithms and their geometric underpinnings

- Spectral, flow, multi-resolution algorithms; their implicit geometric basis; global and scalable local methods; expander-like, tree-like, and hyperbolic structure

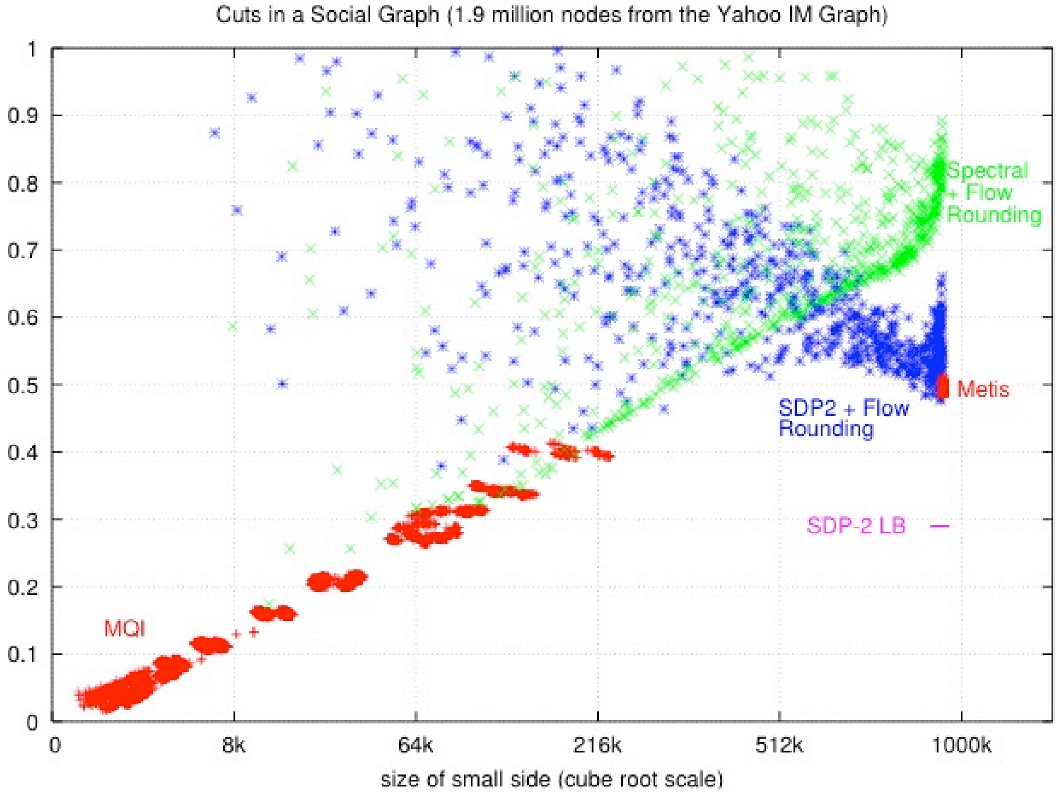
Novel insights on structure in large informatics graphs

- Successes and failures of existing models; empirical results, including “experimental” methodologies for probing network structure, taking into account algorithmic and statistical issues; implications and future directions

An awkward empirical fact

Lang (NIPS 2006), Leskovec, Lang, Dasgupta, and Mahoney (WWW 2008 & arXiv 2008)

Can we cut "internet graphs" into two pieces that are "nice" *and* "well-balanced"?



For *many* **real-world** social-and-information "power-law graphs," there is an *inverse relationship* between "cut quality" and "cut balance."

Consequences of this empirical fact

Relationship b/w small-scale structure and large-scale structure in social/information networks* is not reproduced (even qualitatively) by popular models

• This relationship governs diffusion of information, routing and decentralized search, dynamic properties, etc., etc., etc.

• This relationship also governs (implicitly) the applicability of nearly every common data analysis tool in these apps

*Probably *much* more generally--social/information networks are just so messy and counterintuitive that they provide very good methodological test cases.

Questions of interest ...

What are degree distributions, clustering coefficients, diameters, etc.?

Heavy-tailed, small-world, expander, geometry+rewiring, local-global decompositions, ...

Are there natural clusters, communities, partitions, etc.?

Concept-based clusters, link-based clusters, density-based clusters, ...

(e.g., *isolated* micro-markets with *sufficient money/clicks* with *sufficient coherence*)

How do networks grow, evolve, respond to perturbations, etc.?

Preferential attachment, copying, HOT, shrinking diameters, ...

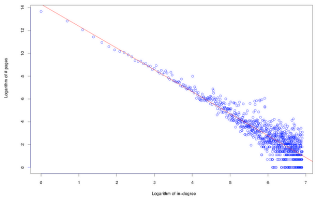
How do dynamic processes - search, diffusion, etc. - behave on networks?

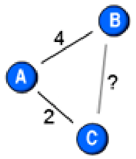
Decentralized search, undirected diffusion, cascading epidemics, ...

How best to do learning, e.g., classification, regression, ranking, etc.?

Information retrieval, machine learning, ...

Popular approaches to network analysis

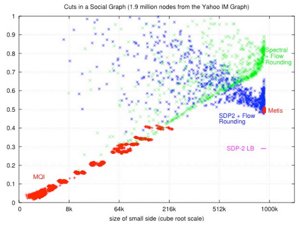




Define simple statistics (clustering coefficient, degree distribution, etc.) and fit simple models

- more complex statistics are too algorithmically complex or statistically rich
- fitting simple stats often doesn't capture what you wanted

Beyond very simple statistics:



- Density, diameter, routing, clustering, communities, …
- *Popular models often fail egregiously at reproducing more subtle properties (even when fit to simple statistics)*

Failings of “traditional” network approaches

Three recent examples of *failings of “small world” and “heavy tailed” approaches*:

• Algorithmic decentralized search - solving a (non-ML) problem: can we find short paths?

• Diameter and density versus time - simple dynamic property

• Clustering and community structure - subtle/complex static property (used in downstream analysis)

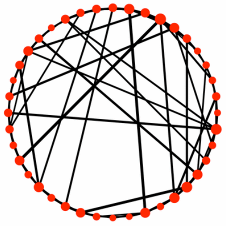
All three examples have to do with the **coupling b/w “local” structure and “global” structure** --- *solution* goes *beyond simple statistics of traditional approaches.*

Failing 1: Search in social graphs

Milgram (1960s)

- Small world experiments - study short paths in social networks
- Individuals from Midwest forward letter to people they know to get it to an individual in Boston.

Watts and Strogatz (1998)



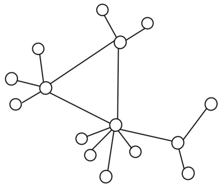
- "Small world" model, i.e., add random edges to an underlying local geometry, reproduces local clustering and existence of short paths

Kleinberg (2000)

- But, even Erdos-Renyi G_{np} random graphs have short paths …
- … so the existence of short paths is not so interesting
- Milgram's experiment also demonstrated people found those paths

Failing 2: Time evolving graphs

Albert and Barabasi (1999)



- "Preferential attachment" model, i.e., at each time step add a constant number of links according to a "rich-get-richer" rule
- Constant average degree, i.e., average node degree remains constant
- Diameter increases roughly logarithmically in time

Leskovec, Kleinberg, and Faloutsos (2005)

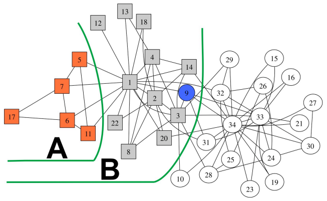
- But, *empirically*, graphs densify over time (i.e., number of edges grows superlinearly with number of nodes) and diameter shrinks over time

Failing 3:
Clustering and community structure

Sociologists (1900s)

- A "community" is any group of two or more people that is useful

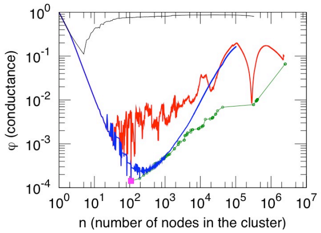
*Girvan and Newman (2002,2004) and **MANY** others*



- A "community" is a set of nodes "joined together in tightly-knit groups between which there are only loose connections

- Modularity becomes a popular "edge counting" metric

Leskovec, Lang, Dasgupta, and Mahoney (2008)

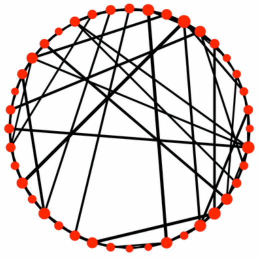


- *All* work on community detection validated on networks with good well-balanced partitions (i.e., low-dimensional and not expanders)

- **But**, *empirically*, larger clusters/communities are less-and-less cluster-like than smaller clusters (i.e., networks are expander-like)

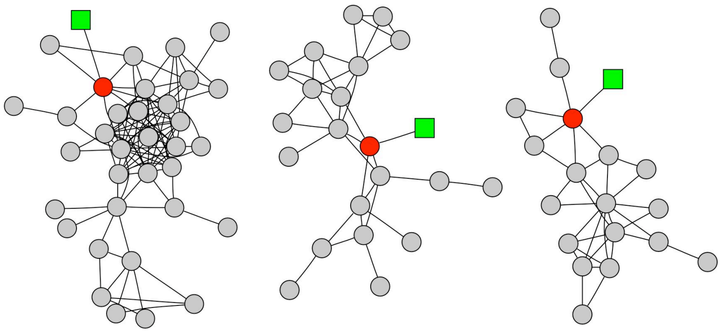
Interplay between preexisting versus generated versus implicit geometry

Preexisting geometry



- Start with geometry and add "stuff"

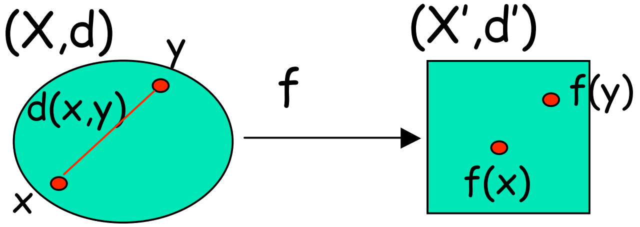
Generated geometry



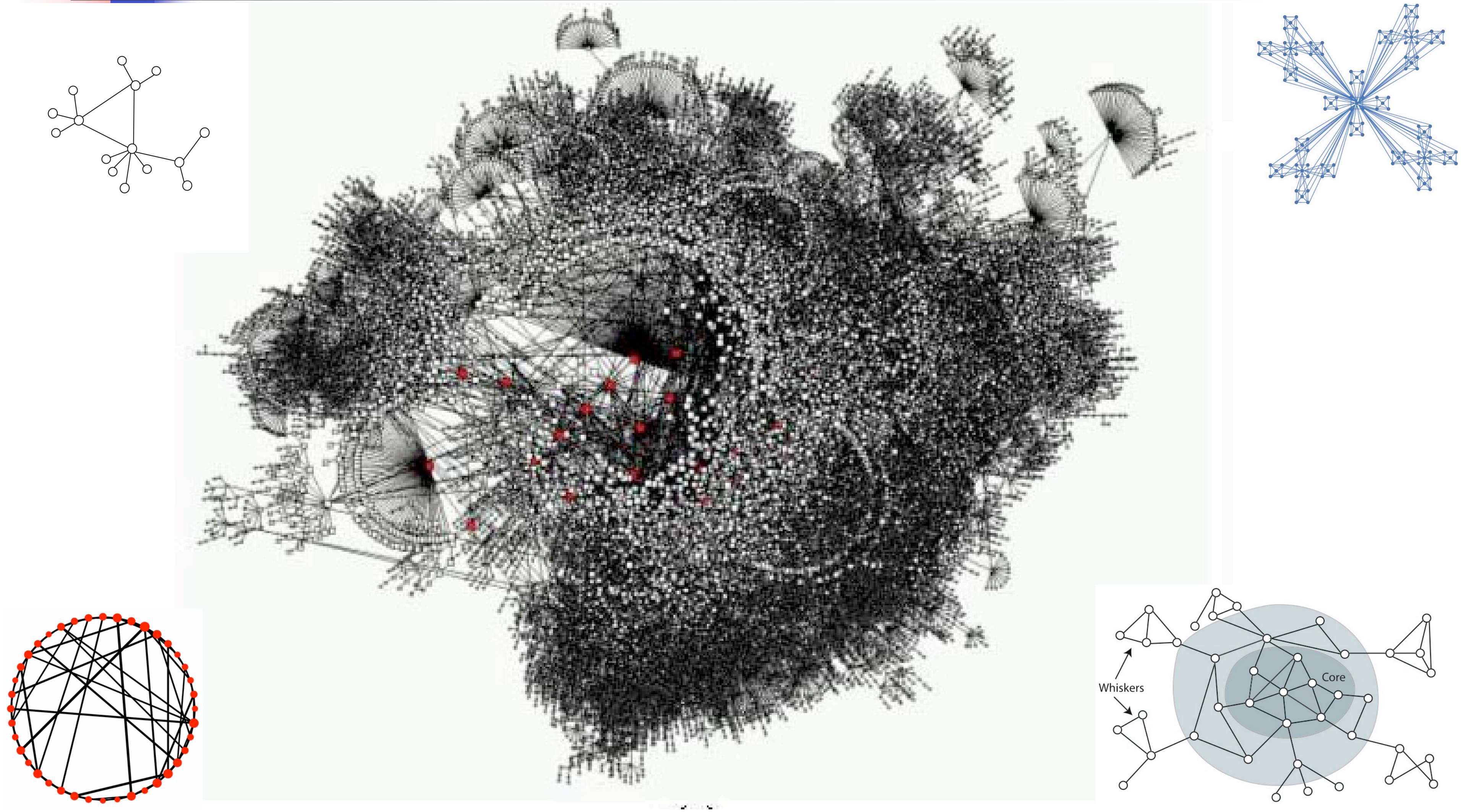
- Generative model leads to structures that are meaningfully-interpretable as geometric

Implicitly-imposed geometry

- Approximation algorithms implicitly embed the data in a metric/geometric place and then round.



What do these networks "look" like?



Approximation algorithms as experimental probes?

Usual *modus operandi* for approximation algorithms for general problems:

- define an objective, the numerical value of which is intractable to compute
- develop approximation algorithm that returns approximation to that number
- graph achieving the approximation may be unrelated to the graph achieving the exact optimum.

But, for randomized approximation algorithms with a geometric flavor (e.g. matrix, regression, eigenvector algorithms; duality algorithms, etc):

- often can approximate the vector achieving the exact solution
- randomized algorithms compute an ensemble of answers -- the details of which depend on choices made by the algorithm
- maybe compare different approximation algorithms for the same problem.

Exptl Tools: Probing Large Networks with Approximation Algorithms

Idea: Use approximation algorithms for NP-hard graph partitioning problems as experimental probes of network structure.

Spectral - (quadratic approx) - confuses "long paths" with "deep cuts"

Multi-commodity flow - (log(n) approx) - difficulty with expanders

SDP - (sqrt(log(n)) approx) - best in theory

Metis - (multi-resolution for mesh-like graphs) - common in practice

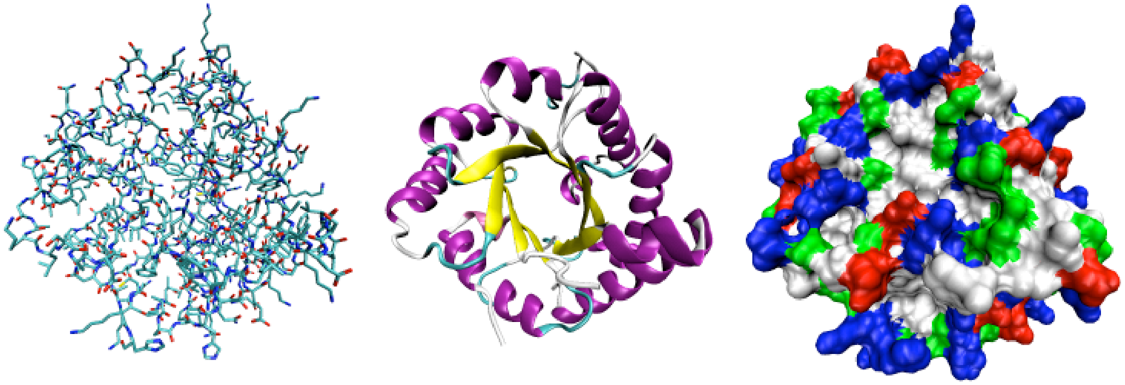
X+MQI - post-processing step on, e.g., Spectral of Metis

Metis+MQI - best conductance (empirically)

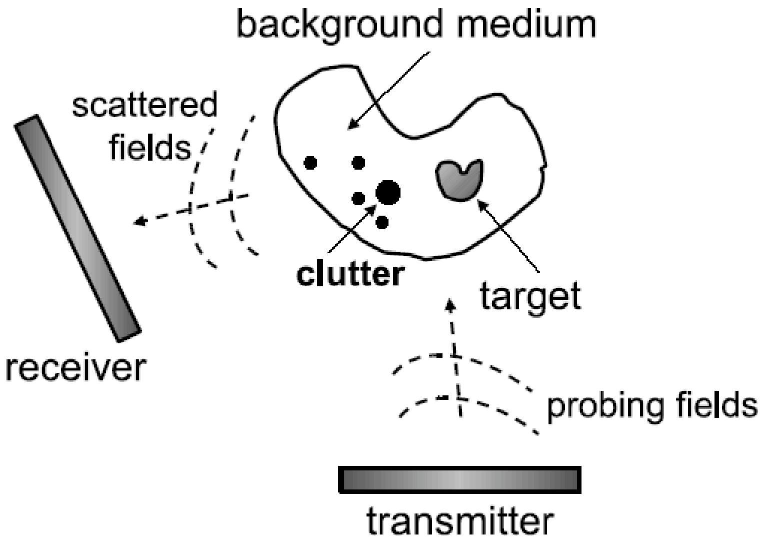
Local Spectral - connected and tighter sets (empirically, regularized communities!)

We are not interested in partitions per se, but in probing network structure.

Analogy: What does a protein look like?



Three possible representations (all-atom; backbone; and solvent-accessible surface) of the three-dimensional structure of the protein triose phosphate isomerase.



Experimental Procedure:

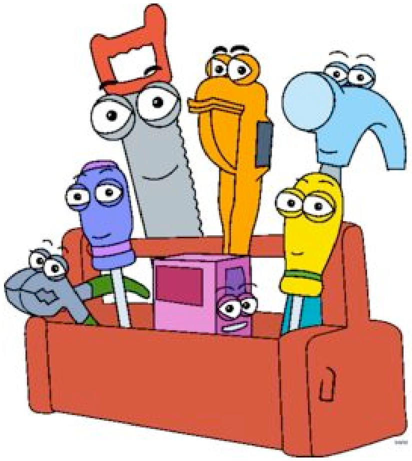
- Generate a bunch of output data by using the unseen object to filter a known input signal.
- Reconstruct the unseen object given the output signal and what we know about the artifactual properties of the input signal.

Experimenting with data with CS tools

- Networks as non-engineered phenomena to be studied as a natural/physical scientist would. (Jon Kleinberg 2006)
- The emergence of cyberspace and the WWW is like the discovery of a new continent. (Jim Gray 1998)
- Want Kepler's Laws of Motion for the Web. (Mike Steuerwalt 1998)

To study data "scientifically," you need

- "Experimental" data (and hopefully lots of it)
- "Experimental" tools (that do the job well)

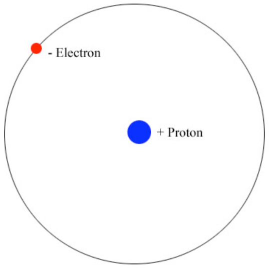


Use approximation algorithms (*and their implicit statistical properties*) as **experimental tools**!

Why graph partitioning? (2 of 2)

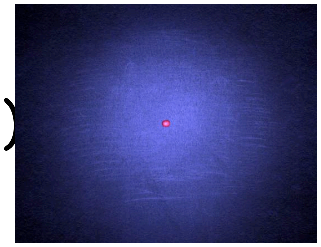
Graph partitioning algorithms:

- tools to "experimentally probe" network structure
- "scalable" and "robust" way to explore extremely non-Euclidean structures in data
- primitive for machine learning and data analysis applications, e.g., image partitioning, semi-supervised learning, etc

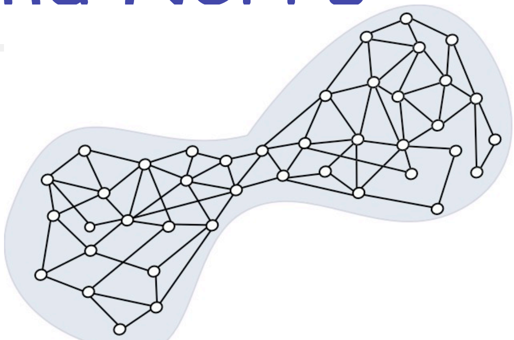


For data more generally:

- "hydrogen atom" for theory/practice disconnect
- "hydrogen atom" for algorithmic vs statistical perspectives
- "hydrogen atom" for regularization implicit in graph algorithms (where you can't "cheat" by data preprocessing)



Communities, Conductance, and NCPPs



Let A be the adjacency matrix of G=(V,E).

The conductance ϕ of a set S of nodes is:

$$\phi(S) = \frac{\sum_{i \in S, j \notin S} A_{ij}}{\min\{A(S), A(\overline{S})\}}$$

$$A(S) = \sum_{i \in S} \sum_{j \in V} A_{ij}$$

The Network Community Profile (NCP) Plot of the graph is:

$$\Phi(k) = \min_{S \subset V, |S|=k} \phi(S)$$

Since algorithms often have non-obvious size-dependent behavior.

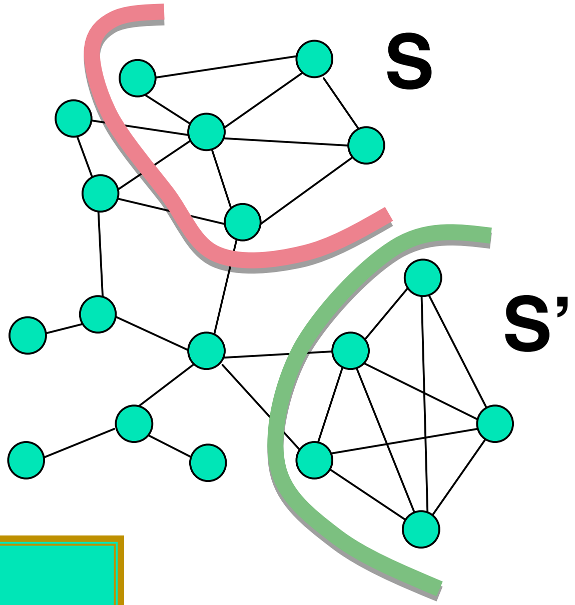
Just as conductance captures the "gestalt" notion of cluster/community quality, the NCP plot measures cluster/community quality ***as a function of size****.*

NCP is intractable to compute --> use approximation algorithms!

Community Score: Conductance

- How community like is a set of nodes?

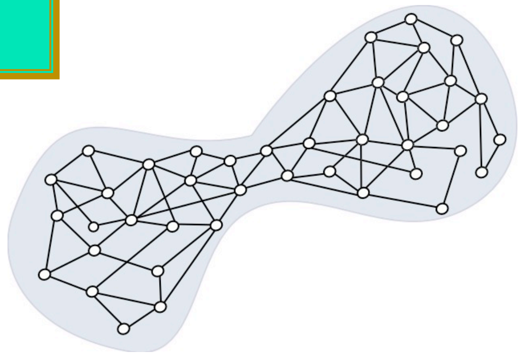
- Need a natural intuitive measure:



- **Conductance** (normalized cut)

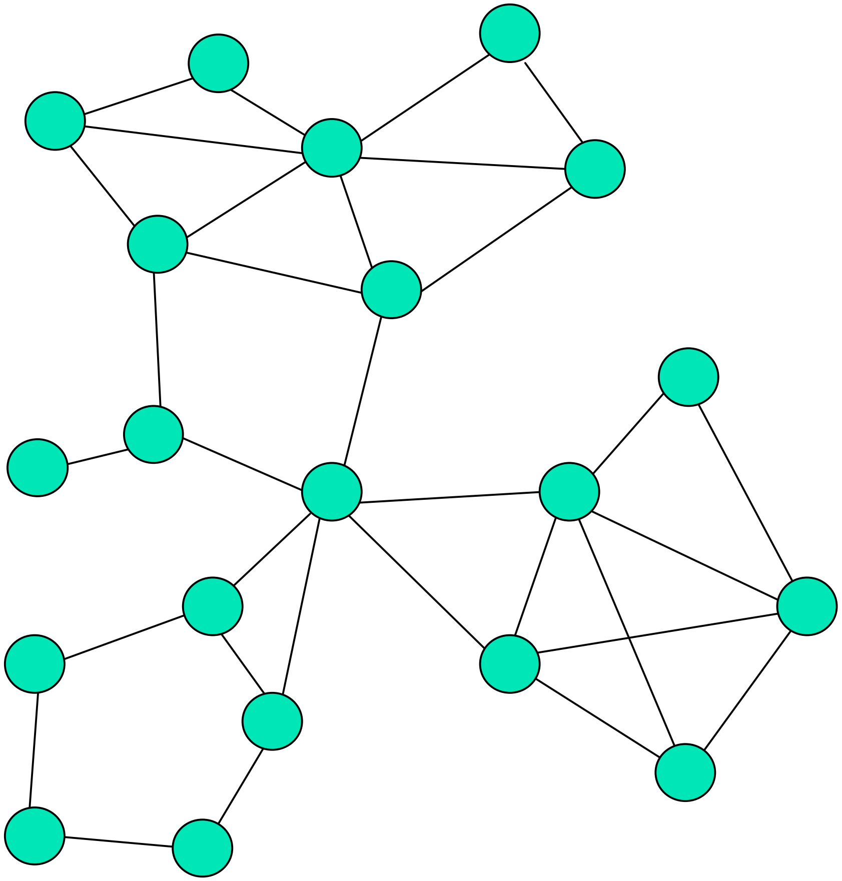
$\phi(S) \approx$ # edges cut / # edges inside

- Small $\phi(S)$ corresponds to more community-like sets of nodes



Community Score: Conductance

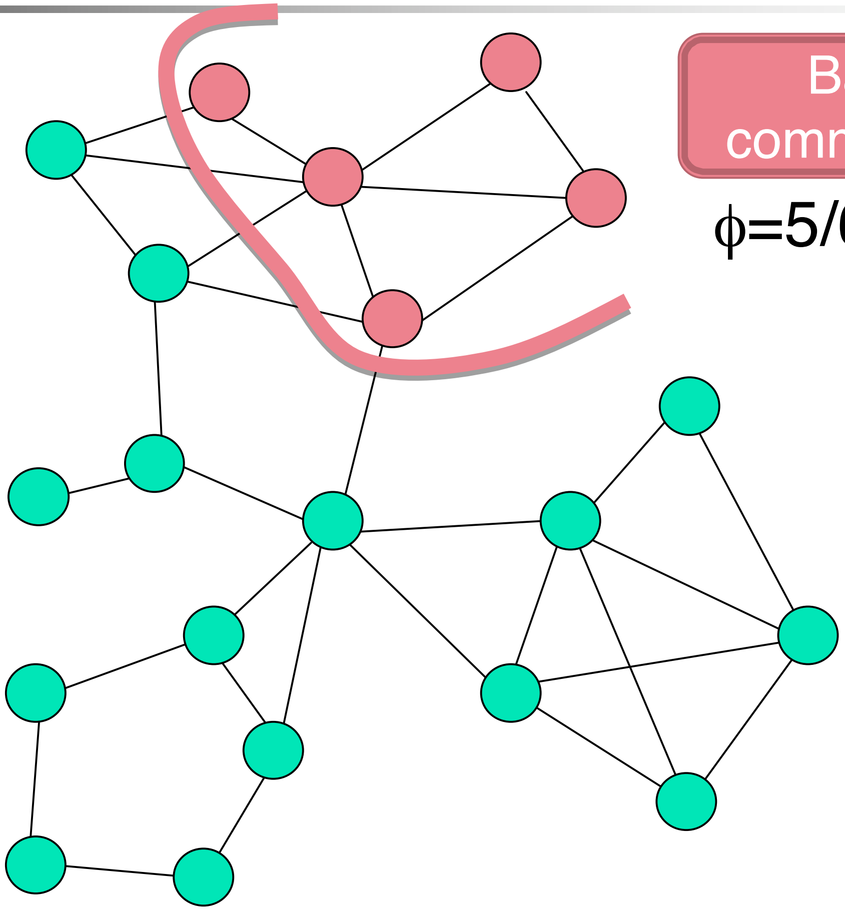
What is "best" community of 5 nodes?



Score: $\phi(S)$ **= # edges cut / # edges inside**

Community Score: Conductance

What is "best" community of 5 nodes?



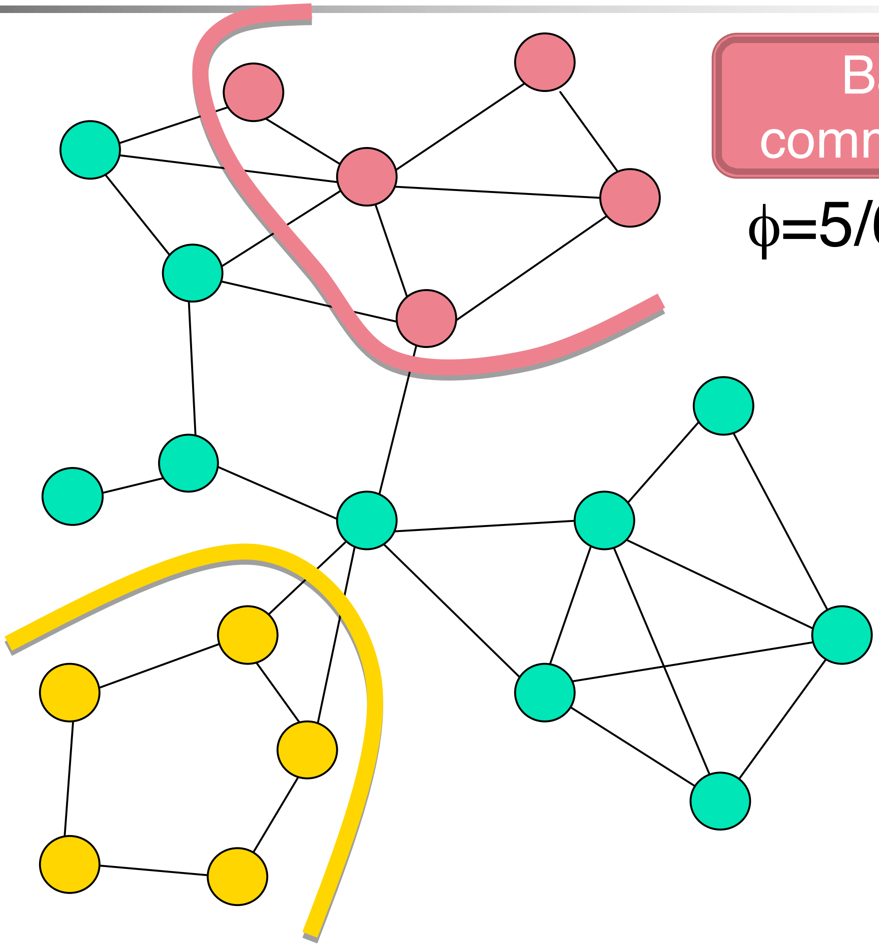
Bad community

ϕ=5/6 = 0.83

Score: ϕ(S) = # edges cut / # edges inside

Community Score: Conductance

What is "best" community of 5 nodes?



Bad community

ϕ=5/6 = 0.83

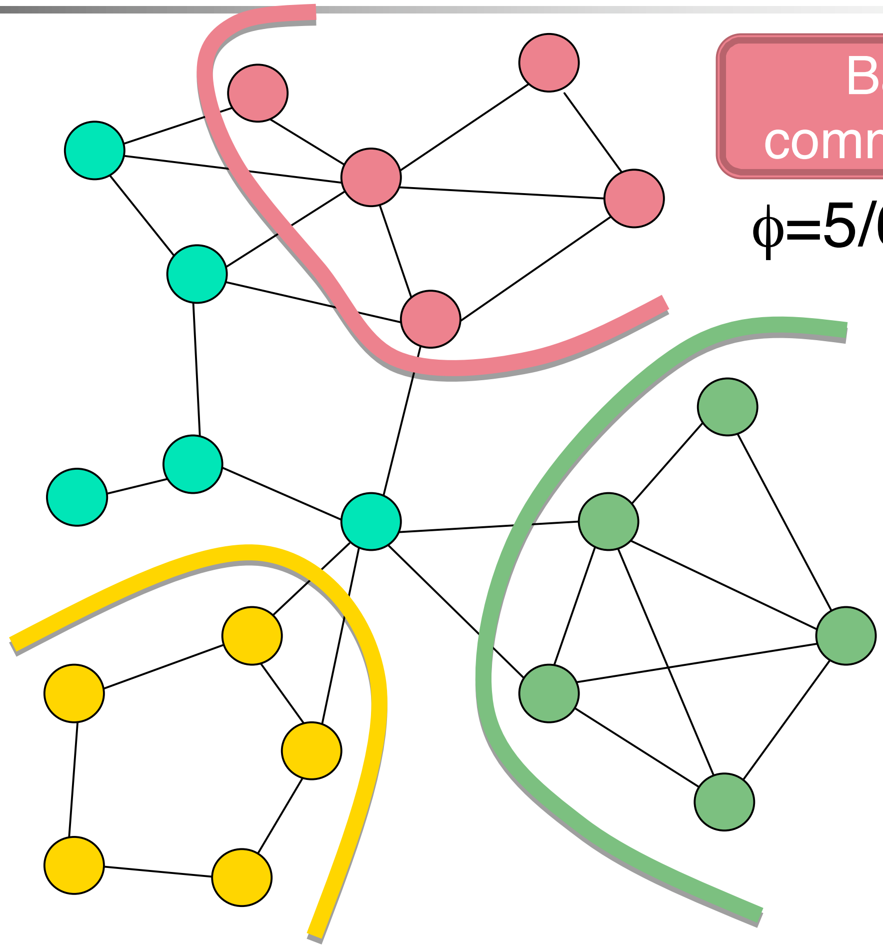
Better community

ϕ=2/5 = 0.4

Score: ϕ(S) = # edges cut / # edges inside

Community Score: Conductance

What is “best” community of 5 nodes?



Bad community

ϕ=5/6 = 0.83

Best community

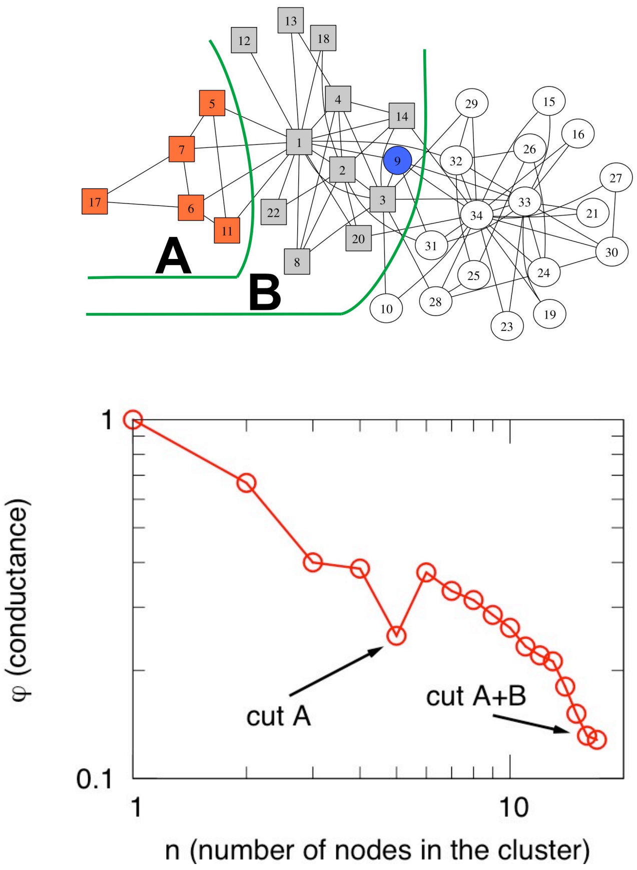
ϕ=2/8 = 0.25

Better community

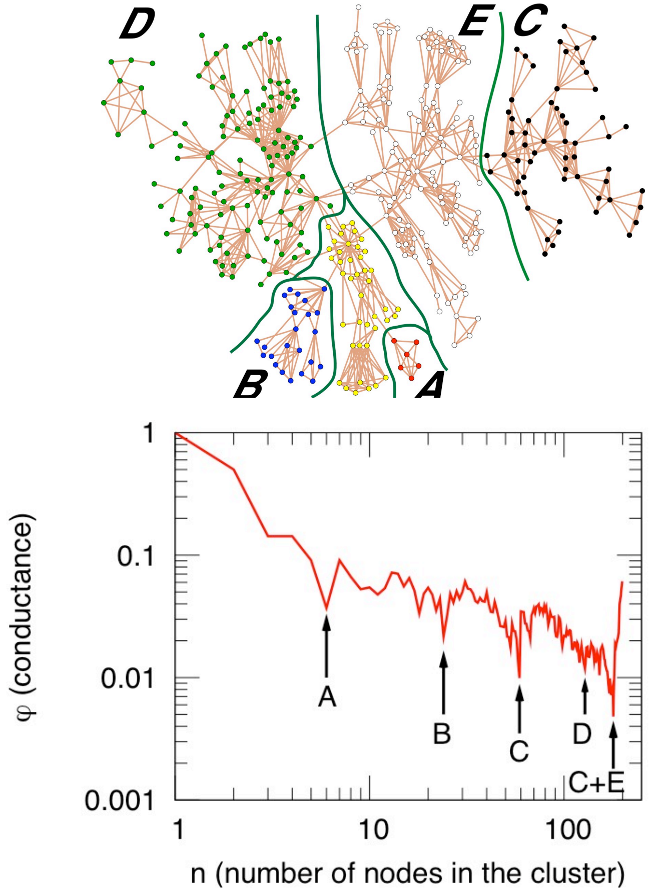
ϕ=2/5 = 0.4

Score: ϕ(S) = # edges cut / # edges inside

Widely-studied small social networks

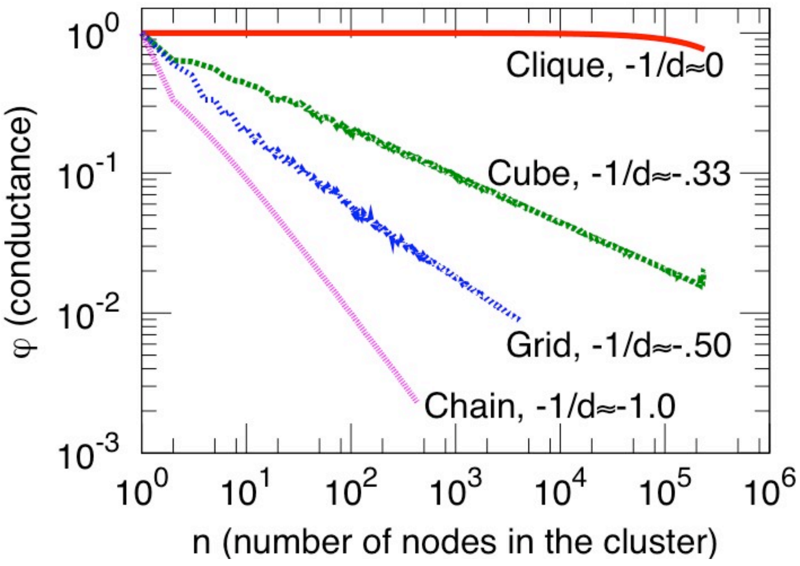


Zachary’s karate club

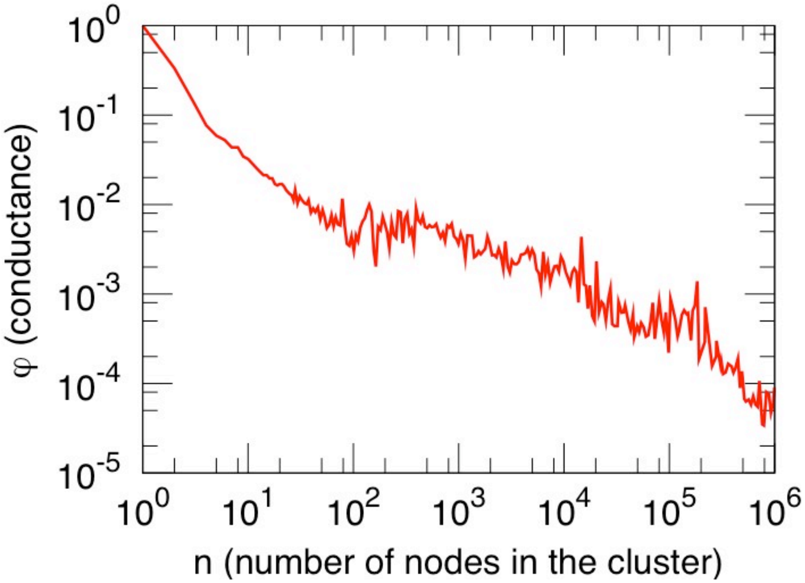


Newman’s Network Science

“Low-dimensional” graphs (and expanders)



d-dimensional meshes



RoadNet-CA

Lots of Generative Models

- Preferential attachment - add edges to high-degree nodes

  (Albert and Barabasi 99, etc.)

- Copying model - add edges to neighbors of a seed node

  (Kumar et al. 00, etc.)

- Hierarchical methods - add edges based on distance in hierarchy

  (Ravasz and Barabasi 02, etc.)

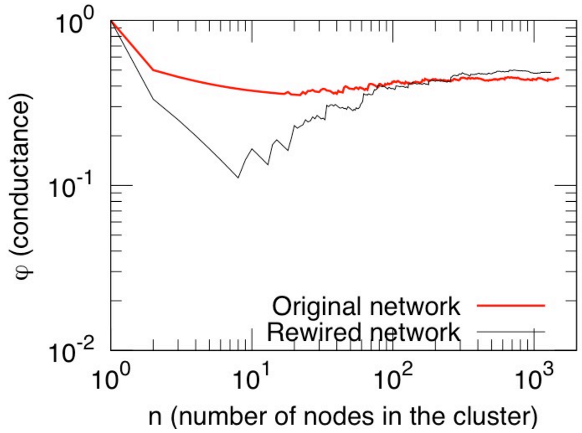
- Geometric PA and Small worlds - add edges to geometric scaffolding

  (Flaxman et al. 04; Watts and Strogatz 98; etc.)

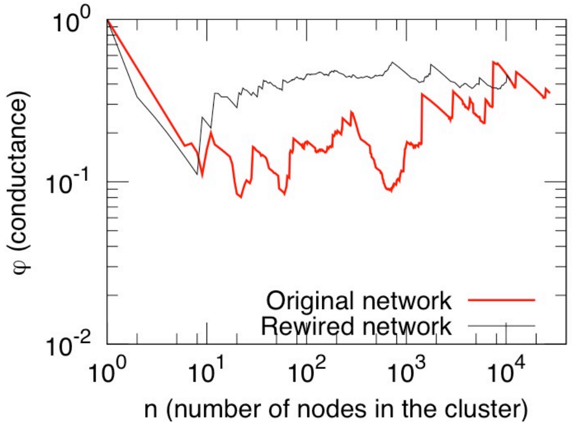
- Random/configuration models - add edges randomly

  (Molloy and Reed 98; Chung and Lu 06; etc.)

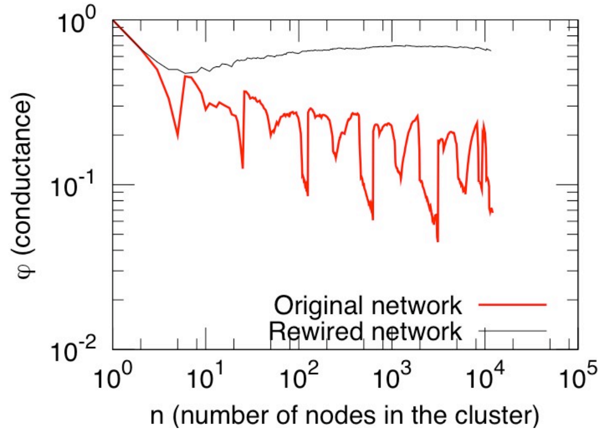
NCPP for common generative models



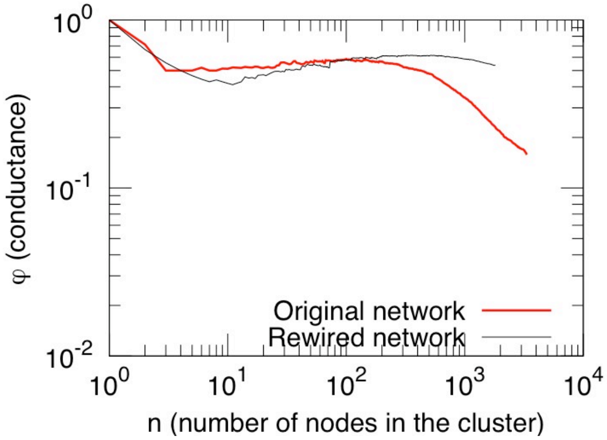
Preferential Attachment



Copying Model

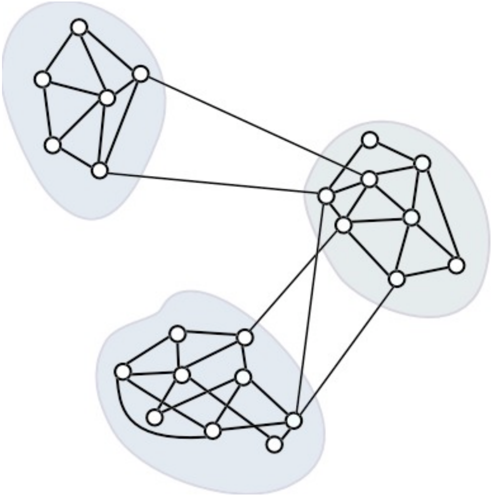


RB Hierarchical



Geometric PA

What do large networks look like?



Downward sloping NCPP

- small social networks (validation)
- “low-dimensional” networks (intuition)
- hierarchical networks (model building)
- existing generative models (incl. community models)

Natural interpretation in terms of isoperimetry

- implicit in modeling with low-dimensional spaces, manifolds, k-means, etc.

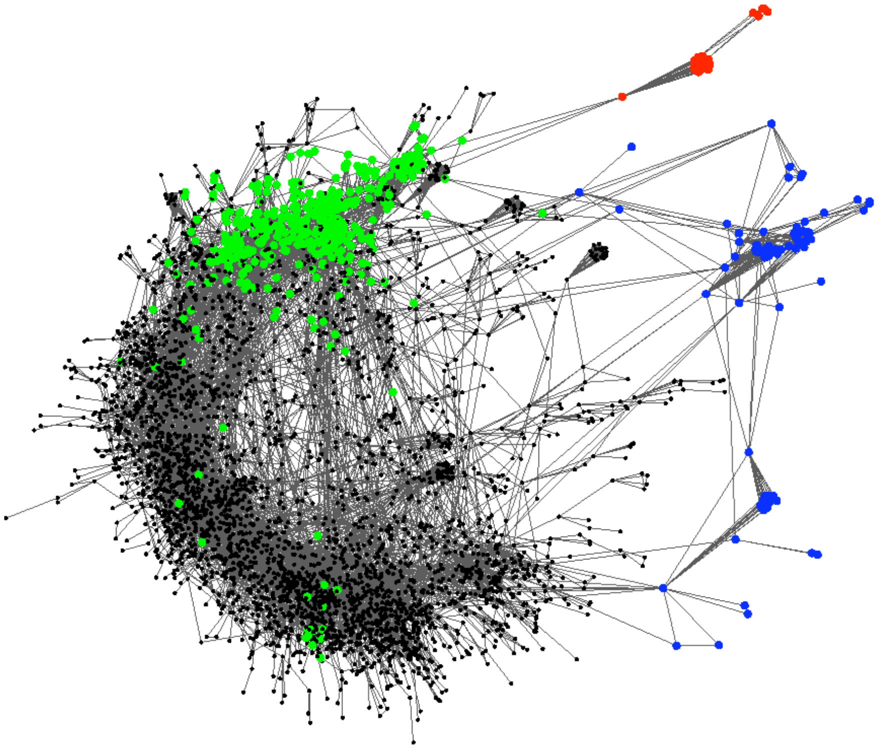
Large social/information networks are very *very* different

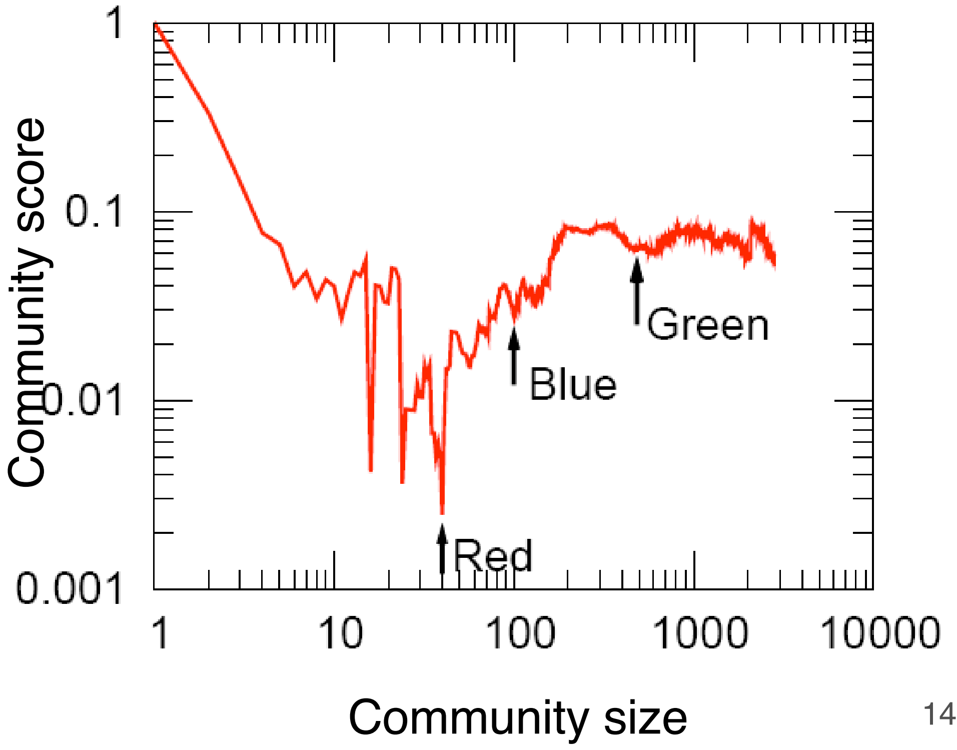
- We examined more than 70 large social and information networks
- We developed principled methods to interrogate large networks
- Previous community work: on small social networks (hundreds, thousands)

Typical example of our findings

Leskovec, Lang, Dasgupta, and Mahoney (WWW 2008 & arXiv 2008)

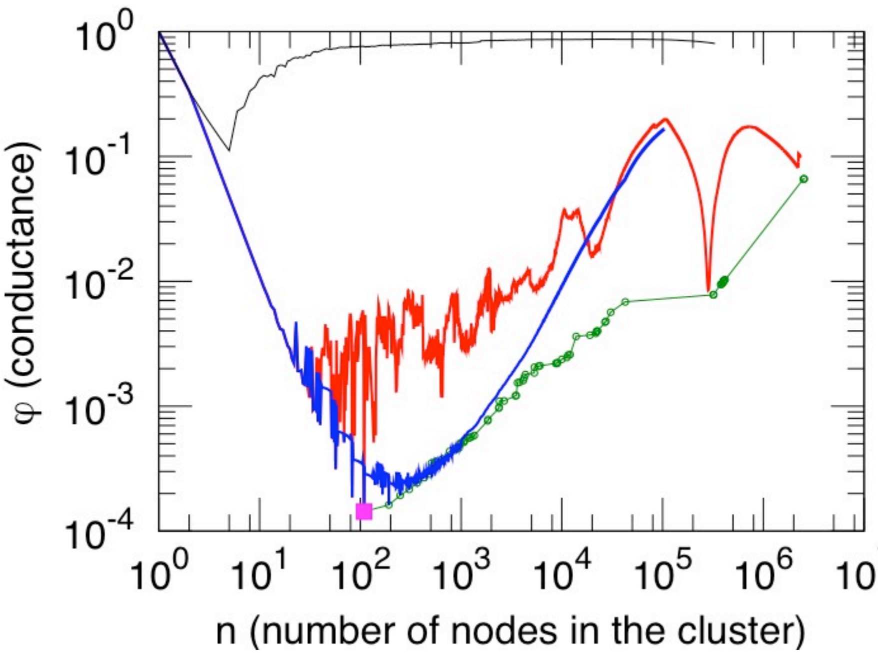
General relativity collaboration network
(4,158 nodes, 13,422 edges)



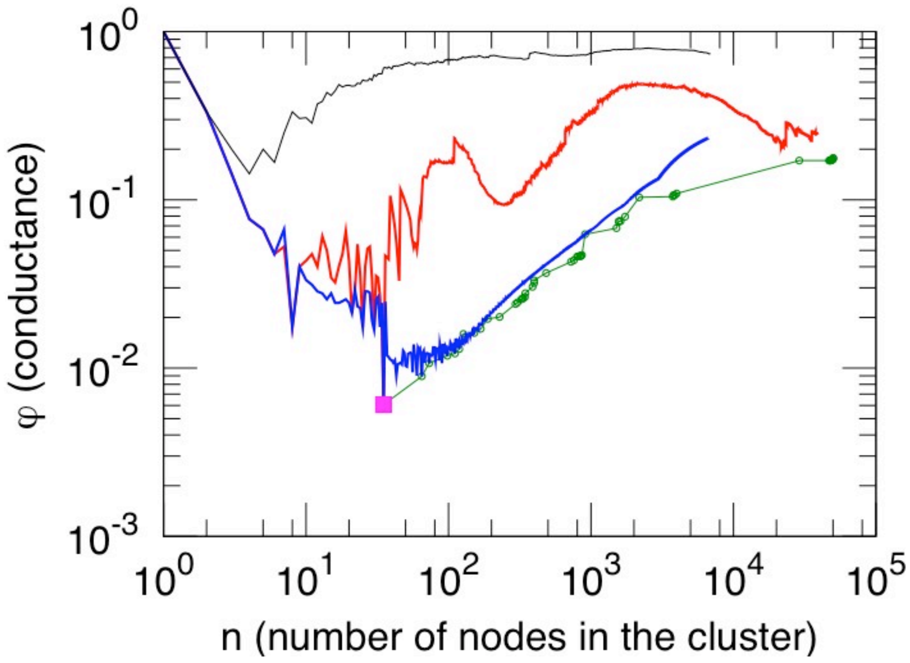


Large Social and Information Networks

Leskovec, Lang, Dasgupta, and Mahoney (WWW 2008 & arXiv 2008)



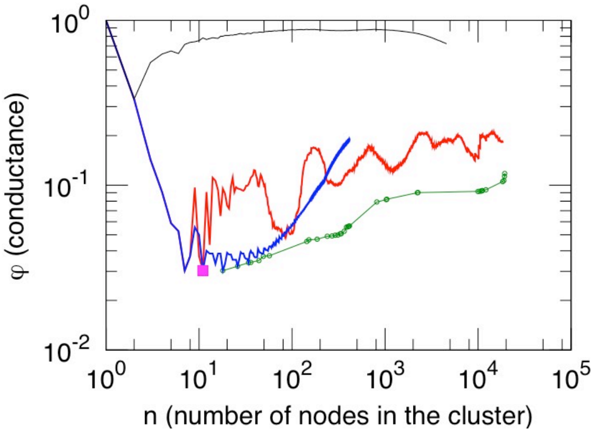
LiveJournal



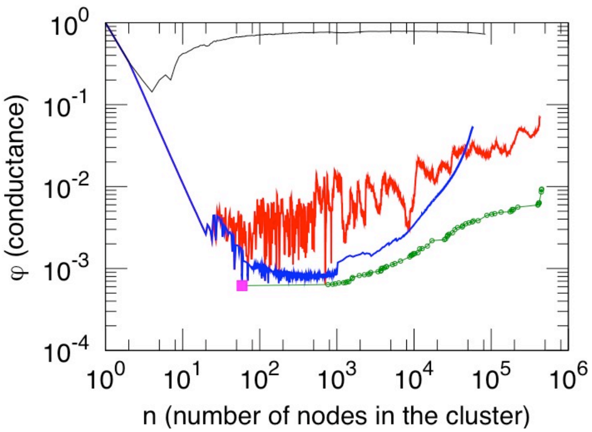
Epinions

Focus on the red curves (local spectral algorithm) - blue (Metis+Flow), green (Bag of whiskers), and black (randomly rewired network) for consistency and cross-validation.

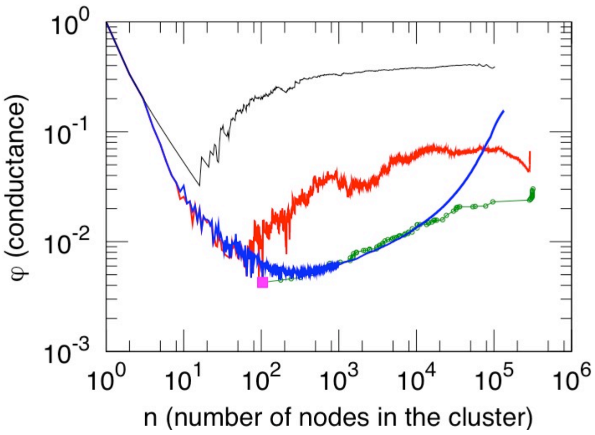
More large networks



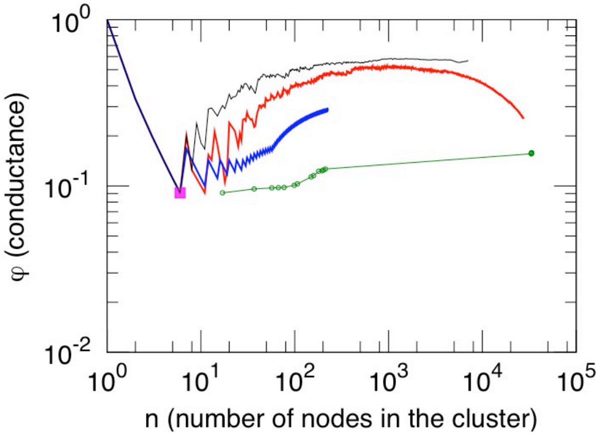
Cit-Hep-Th



Web-Google

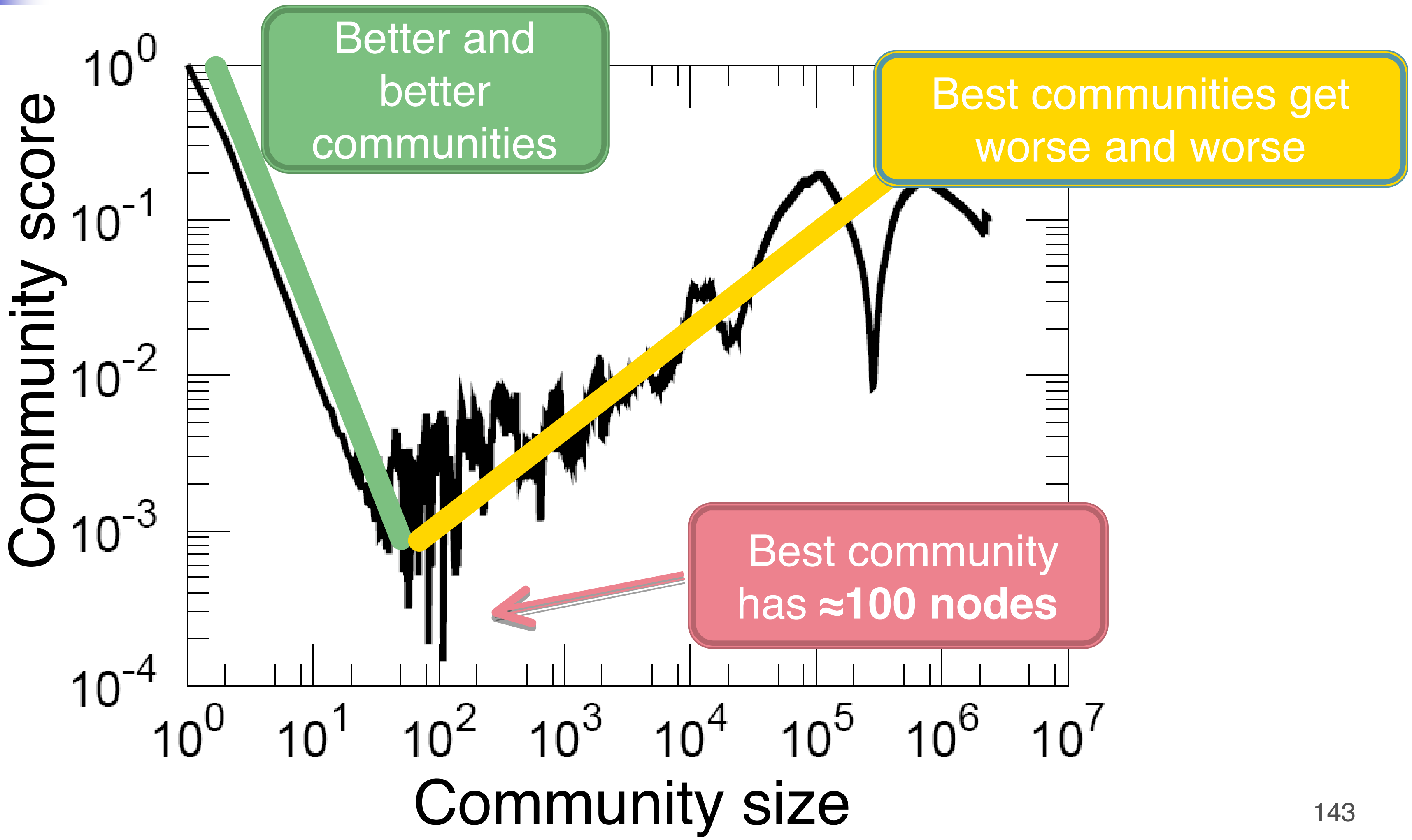


AtP-DBLP



Gnutella

NCPP: LiveJournal (N=5M, E=43M)



Better and better communities

Best communities get worse and worse

Best community has **≈100 nodes**

How do we know this plot it “correct”?

- Algorithmic Result

  Ensemble of sets returned by different algorithms are very different

  Spectral vs. flow vs. bag-of-whiskers heuristic

- Statistical Result

  Spectral method implicitly regularizes, gets more meaningful communities

- Lower Bound Result

  Spectral and SDP lower bounds for large partitions

- Structural Result

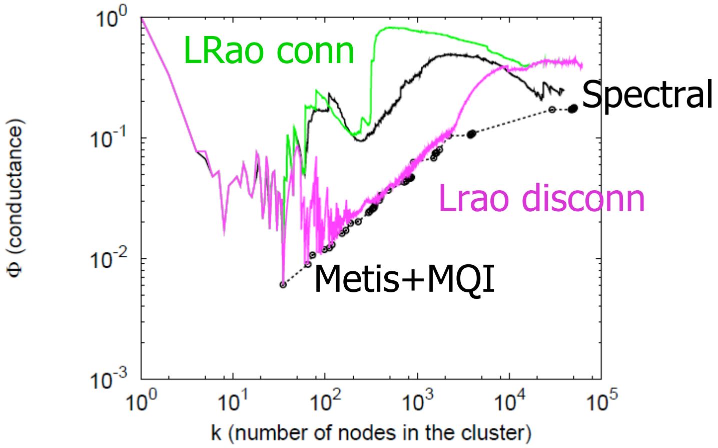
  Small barely-connected “whiskers” responsible for minimum

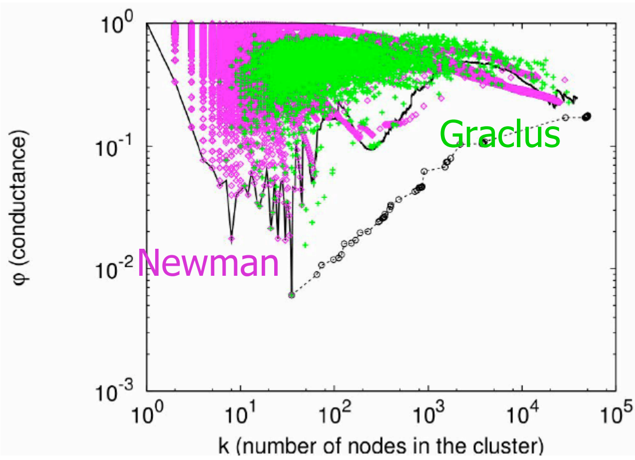
- Modeling Result

  Very sparse Erdos-Renyi (or PLRG wth β ε (2,3)) gets imbalanced deep cuts

Other clustering methods

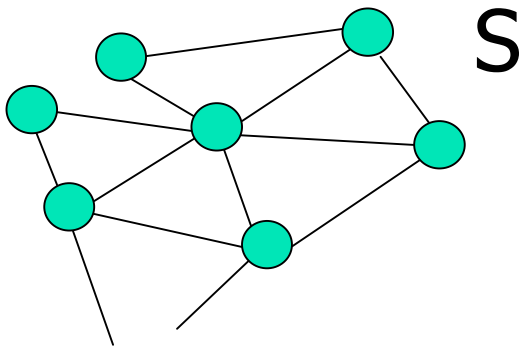
- LeightonRao: based on multi-commodity flow
  - Disconnected clusters vs. Connected clusters
- Graclus prefers larger clusters
- Newman's modularity optimization similar to Local Spectral





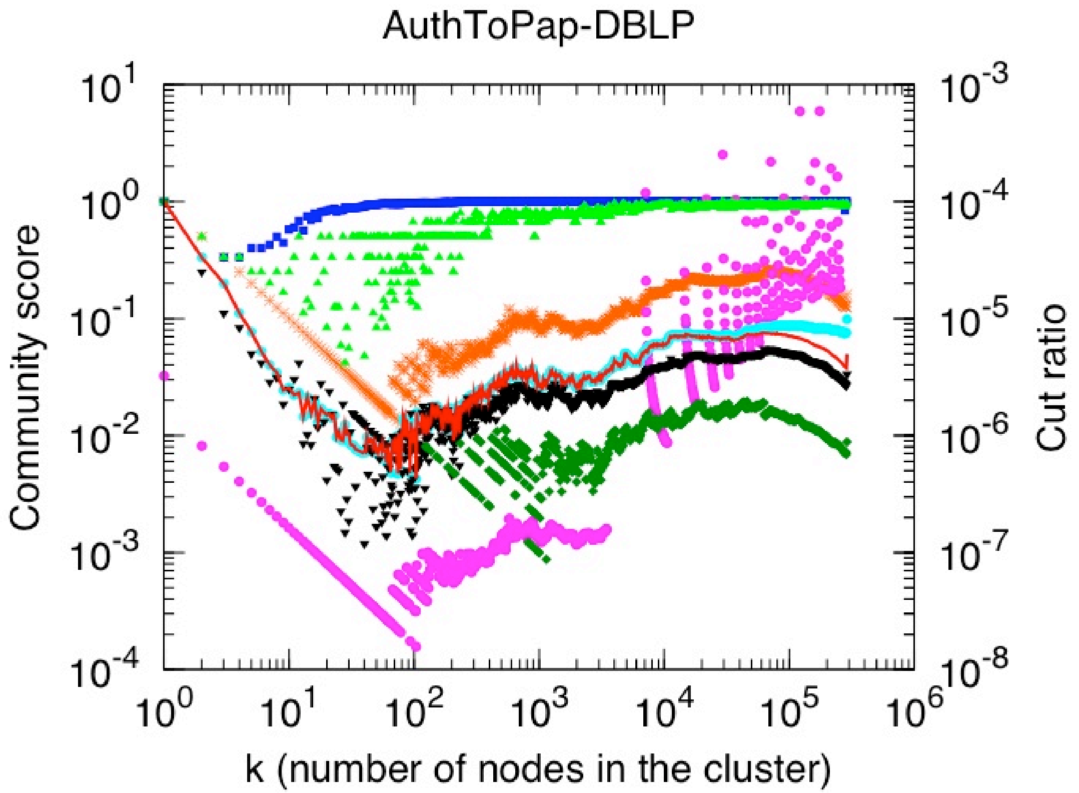
12 objective functions

- Clustering objectives:
  - Single-criterion:
    - Modularity: *m-E(m)* *(Volume minus correction)*
    - Modularity Ratio: *m-E(m)*
    - Volume: $\sum_u d(u)=2m+c$
    - Edges cut: *c*
  - Multi-criterion:
    - Conductance: *c/(2m+c)* *(SA to Volume)*
    - Expansion: *c/n*
    - Density: $1\text{-}m/n^2$
    - CutRatio: *c/n(N-n)*
    - Normalized Cut: *c/(2m+c) + c/2(M-m)+c*
    - Max ODF: *max frac. of edges of a node pointing outside S*
    - Average-ODF: *avg. frac. of edges of a node pointing outside*
    - Flake-ODF: *frac. of nodes with mode than _ edges inside*



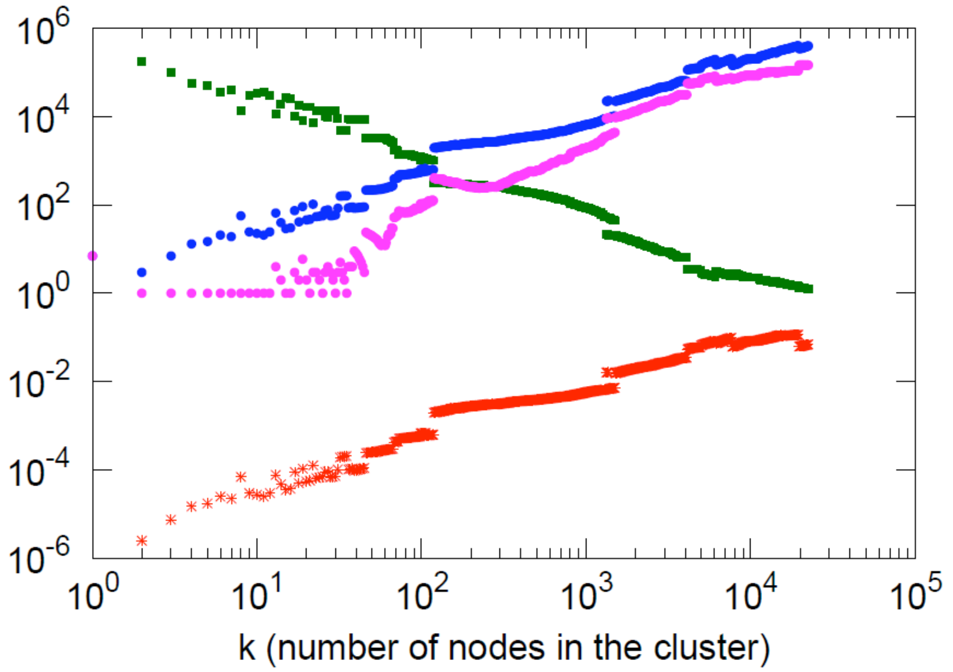
n: nodes in S
m: edges in S
c: edges pointing
outside S

Multi-criterion objectives



- Qualitatively similar to conductance
- Observations:
  - Conductance, Expansion, NCut, Cut-ratio and Avg-ODF are similar
  - Max-ODF prefers smaller clusters
  - Flake-ODF prefers larger clusters
  - Internal density is bad
  - Cut-ratio has high variance

Single-criterion objectives

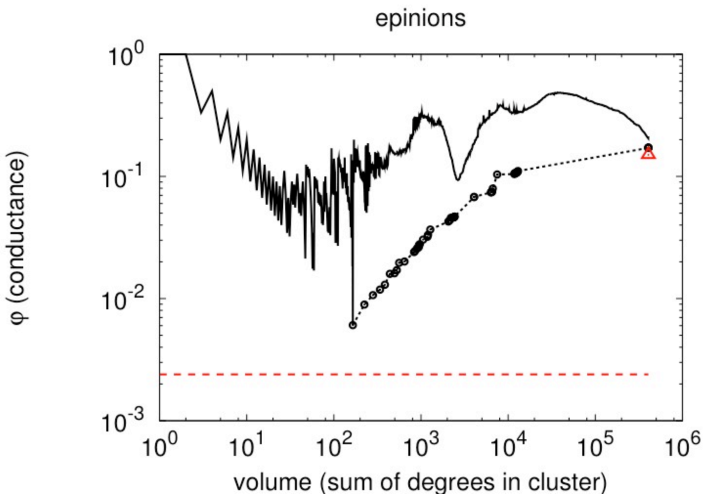


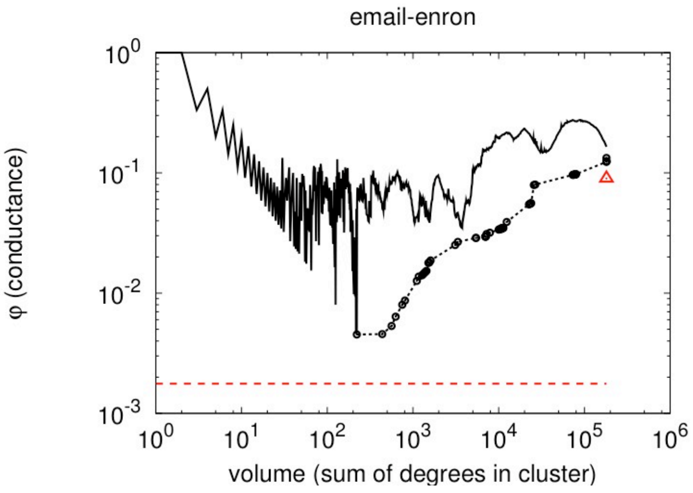
Observations:

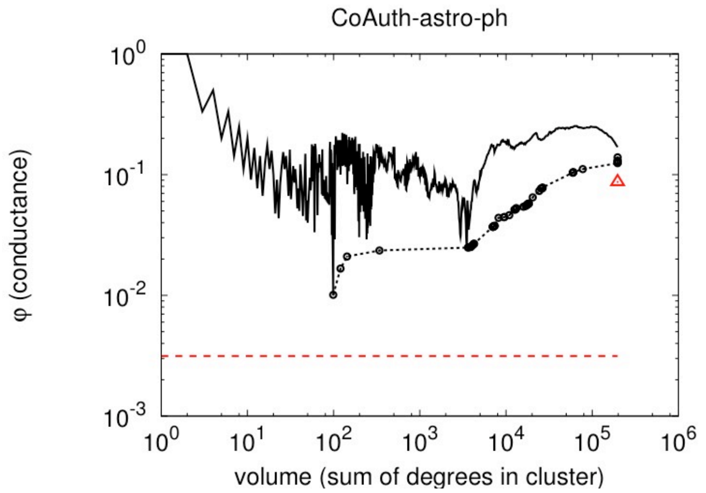
- All measures are monotonic (for rather trivial reasons)
- Modularity
  - prefers large clusters
  - Ignores small clusters
  - *Because it basically captures Volume!*

Lower and upper bounds

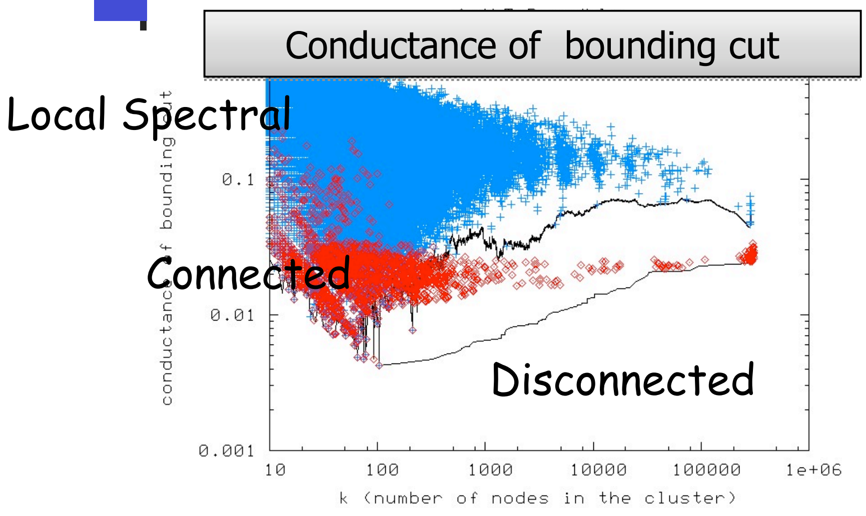
- Lower bounds on conductance can be computed from:
  - Spectral embedding (independent of balance)
  - SDP-based methods (for volume-balanced partitions)
- Algorithms find clusters close to theoretical lower bounds

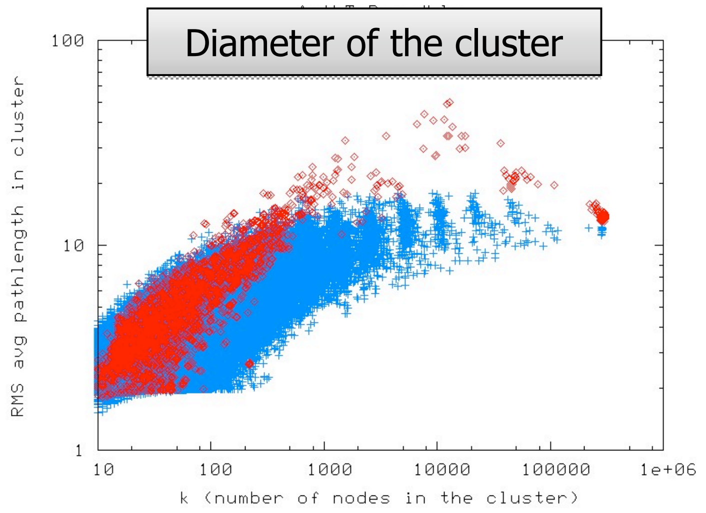


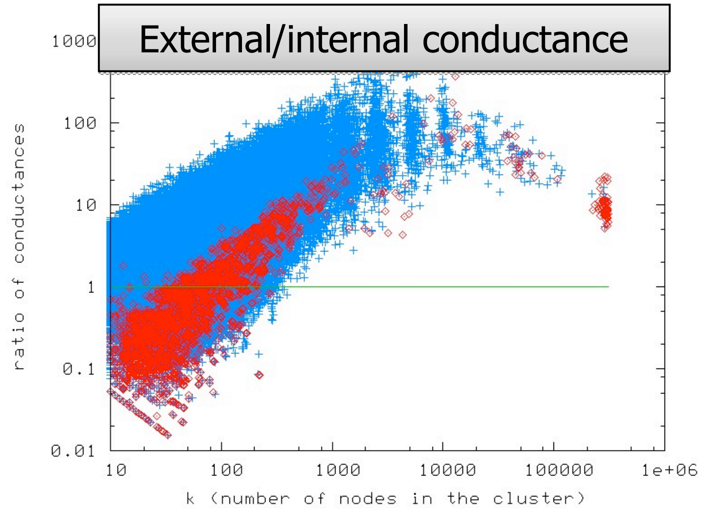




Regularized and non-regularized communities (1 of 2)





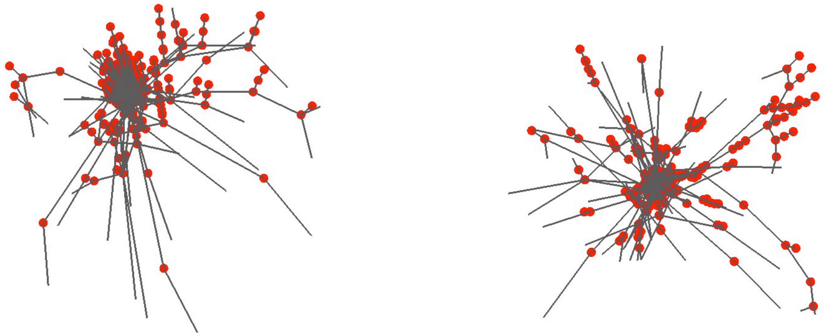


Lower is good

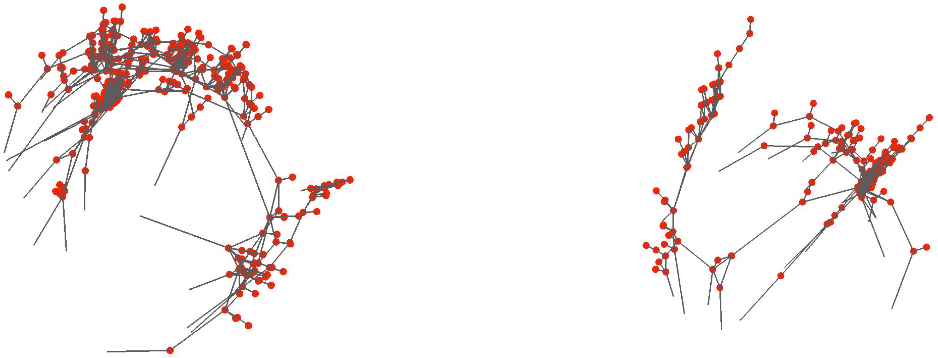
- Metis+MQI (red) gives sets with better conductance.
- Local Spectral (blue) gives tighter and more well-rounded sets.

Regularized and non-regularized communities (2 of 2)

Two ca. 500 node communities from Local Spectral Algorithm:



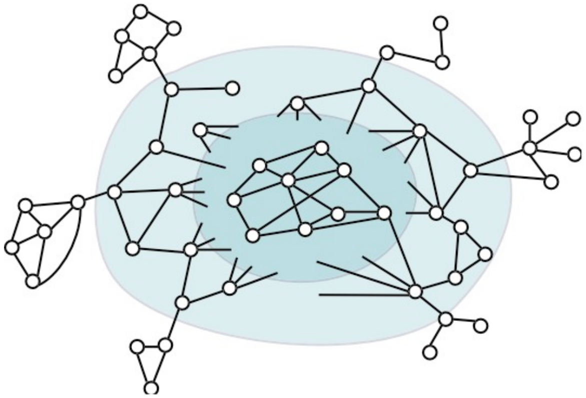
Two ca. 500 node communities from Metis+MQI:

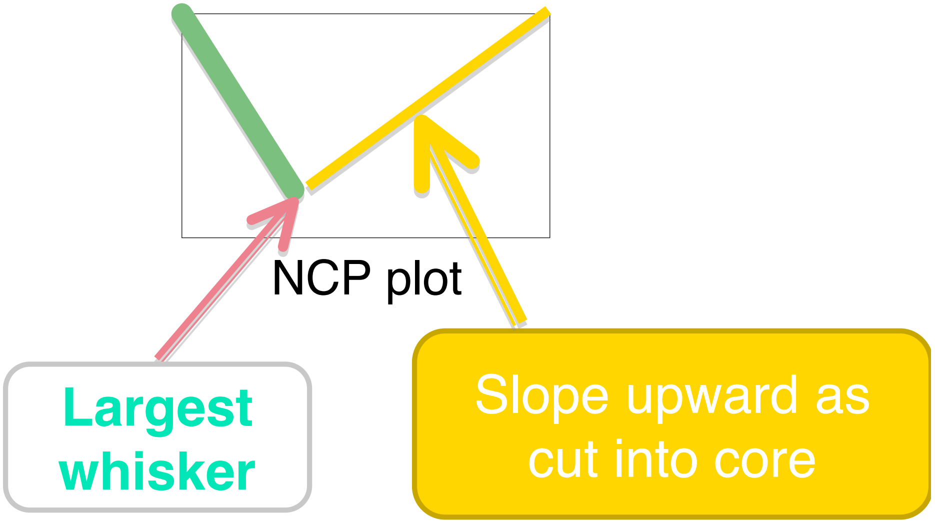


Interpretation: "Whiskers" and the "core" of large informatics graphs

- "Whiskers"
  - maximal sub-graph detached from network by removing a single edge
  - contains 40% of nodes and 20% of edges
- "Core"
  - the rest of the graph, i.e., the 2-edge-connected core

- Global minimum of NCPP is a whisker
- BUT, *core itself has nested whisker-core structure*





NCP plot

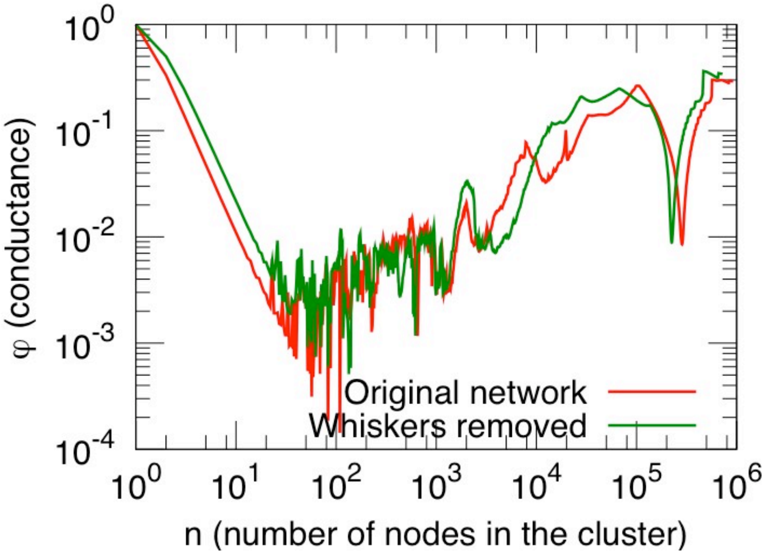
Largest whisker

Slope upward as cut into core

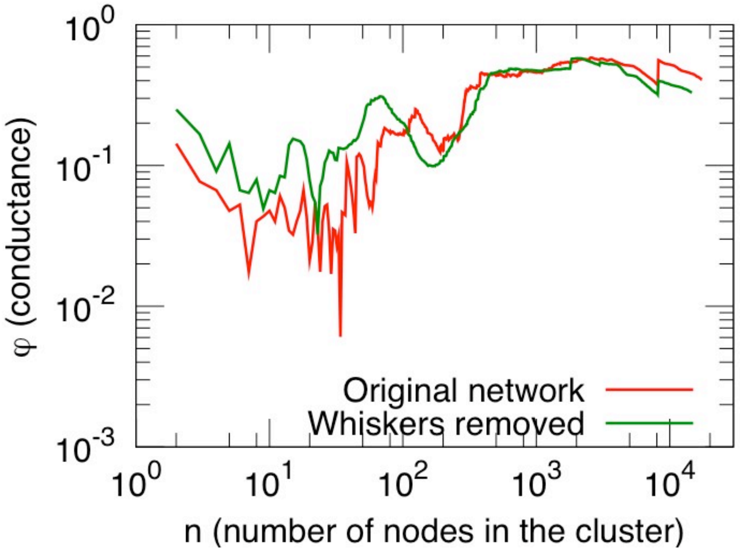
What if the "whiskers" are removed?

Then the lowest conductance sets - the "best" communities - are "2-whiskers."

(So, the "core" peels apart like an onion.)

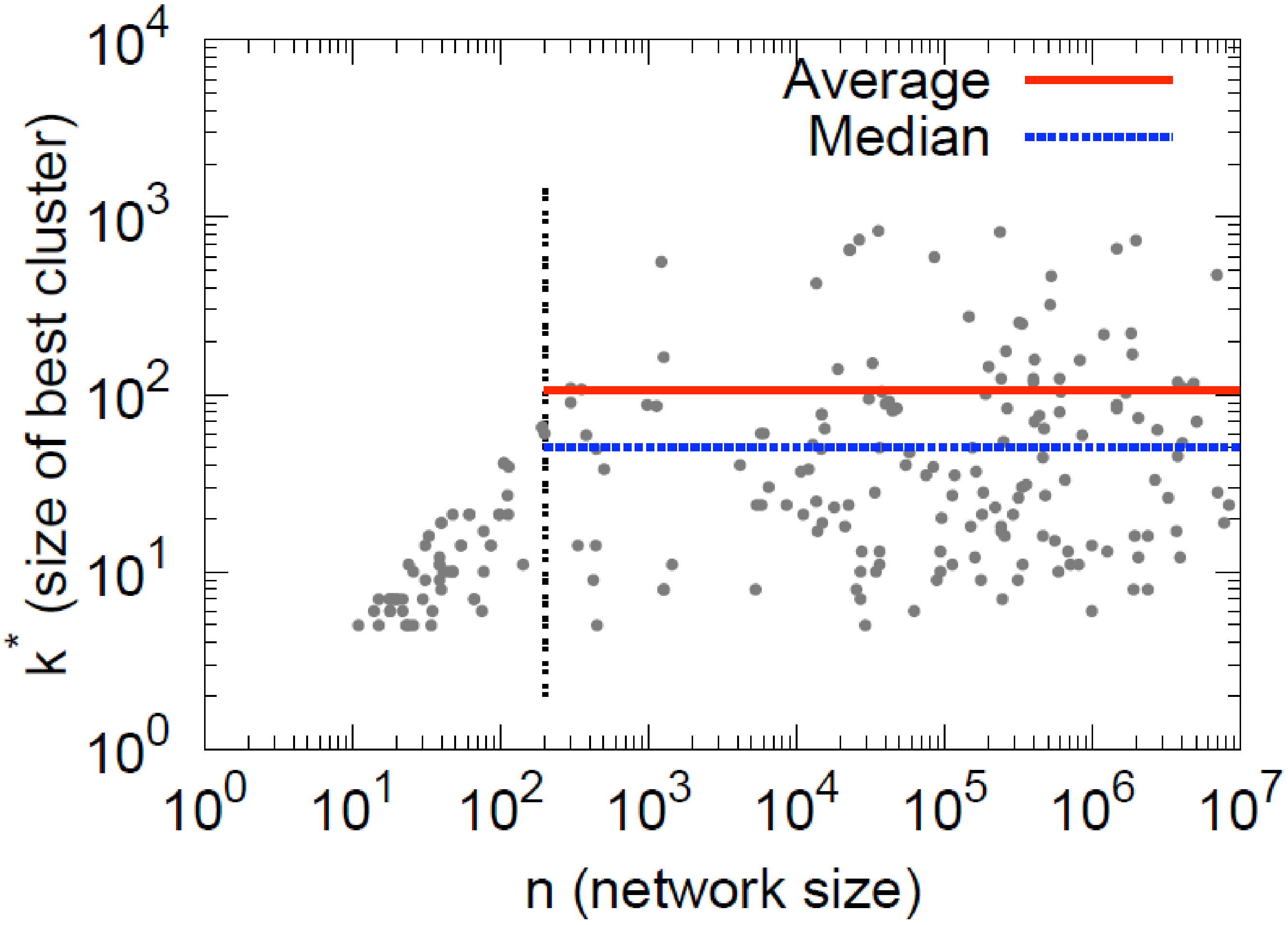


LiveJournal



Epinions

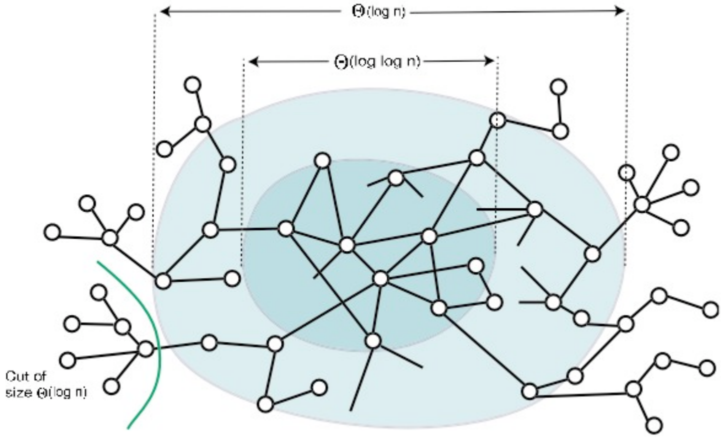
Size of best cluster versus network size



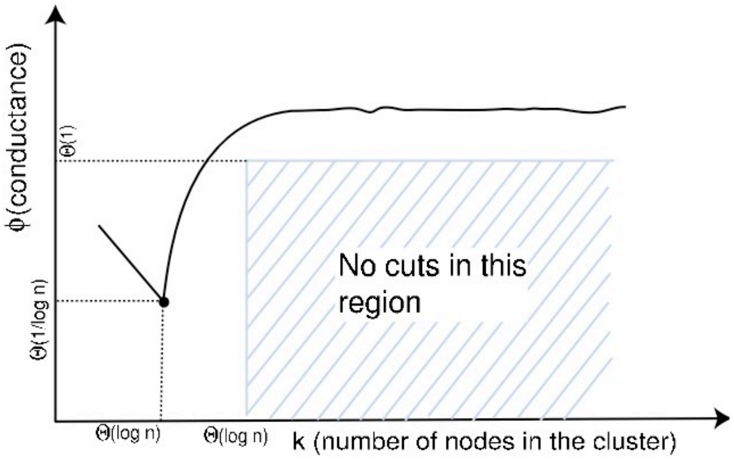
(Each dot is a different network -- so they are roughly independent.)

Interpretation: A simple theorem on random graphs

Let $\mathbf{w} = (w_1, \ldots, w_n)$, where
$w_i = ci^{-1/(\beta-1)}, \quad \beta \in (2,3).$
Connect nodes i and j w.p.
$p_{ij} = w_i w_j / \sum_k w_k.$



Structure of the G(w) model, with β ε (2,3).



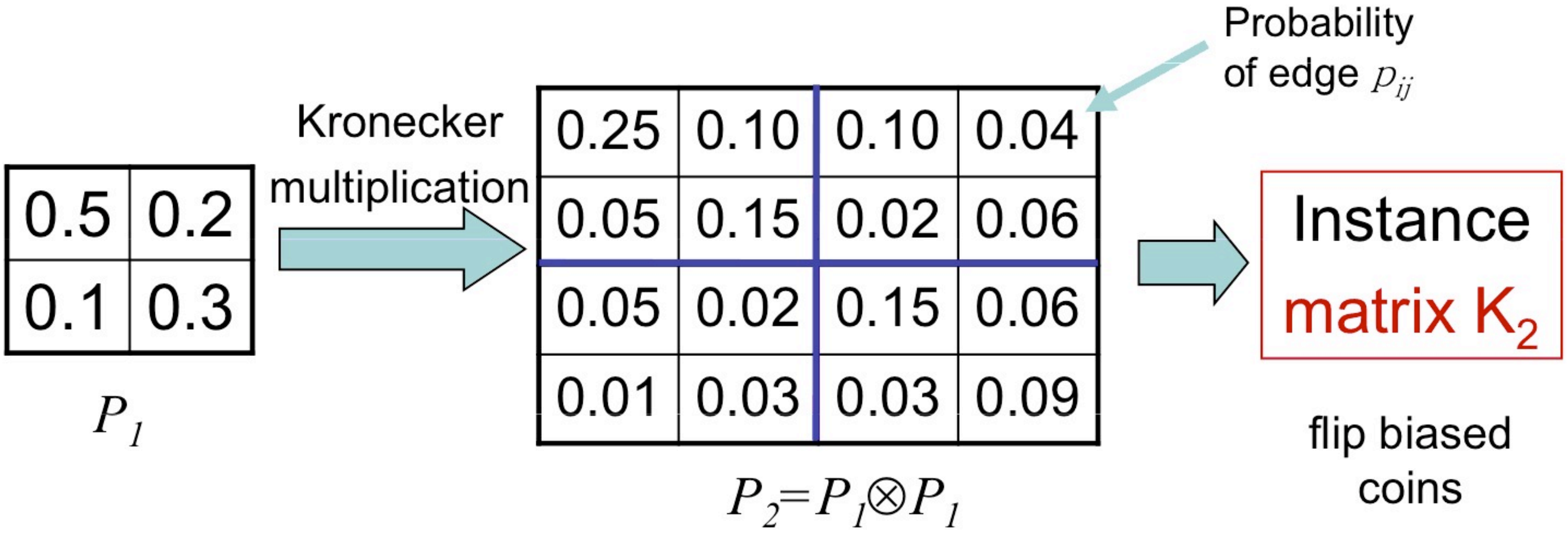
Power-law random graph with β ε (2,3).

- **Sparsity** (coupled with randomness) **is the issue**, *not* heavy-tails.
- (Power laws with β ε (2,3) give us the appropriate sparsity.)

α	β
β	γ

Stochastic Kronecker Graphs

Leskovec, et al. (arXiv 2009); Mahdian-Xu 2007



0.5	0.2
0.1	0.3

P_1

0.25	0.10	0.10	0.04
0.05	0.15	0.02	0.06
0.05	0.02	0.15	0.06
0.01	0.03	0.03	0.09

$P_2=P_1 \otimes P_1$

Deterministic version - can reproduce HT degrees, densification power law, etc

Stochastic version - Ass $1 \geq \alpha \geq \beta \geq \gamma \geq 0$. Connected iff $\beta+\gamma>1$ or $\alpha=\beta=1,\gamma=0$. Giant component iff $(\alpha+\beta)(\beta+\gamma)>1$ or $(\alpha+\beta)(\beta+\gamma)=1$, $\alpha+\beta>\beta+\gamma$

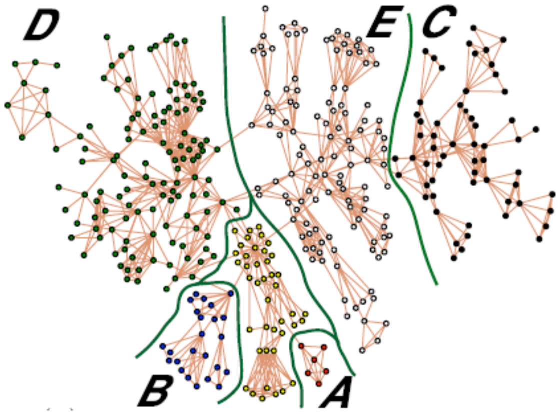
$$\begin{array}{|c|c|}\hline \alpha & \beta \\ \hline \beta & \gamma \\ \hline \end{array}$$

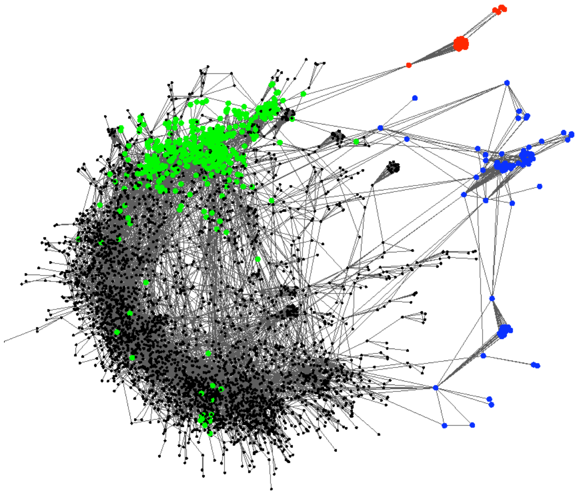
Small versus Large Networks

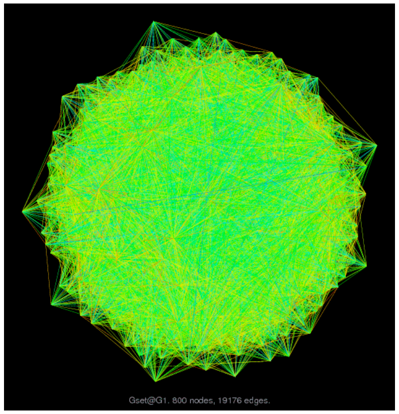
Leskovec, et al. (arXiv 2009); Mahdian-Xu 2007

- Small and large networks are very different:

(also, an expander)







E.g., fit these networks to Stochastic Kronecker Graph with "base" K=[a b; b c]:

$$K_1 = \begin{array}{|c|c|}\hline 0.99 & 0.17 \\ \hline 0.17 & 0.82 \\ \hline \end{array}$$

$$\begin{array}{|c|c|}\hline 0.99 & 0.55 \\ \hline 0.55 & 0.15 \\ \hline \end{array}$$

$$\begin{array}{|c|c|}\hline 0.2 & 0.2 \\ \hline 0.2 & 0.2 \\ \hline \end{array}$$

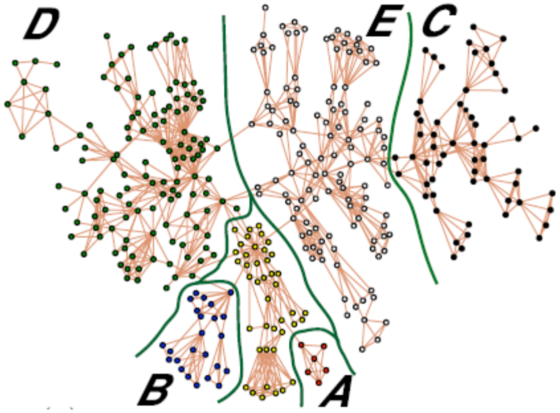
α	β
β	γ

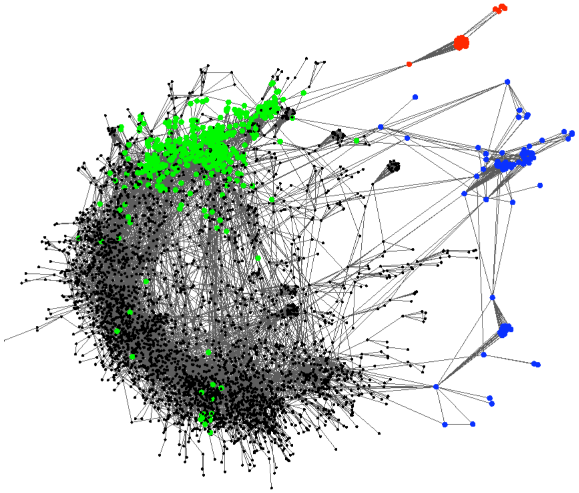
Small versus Large Networks

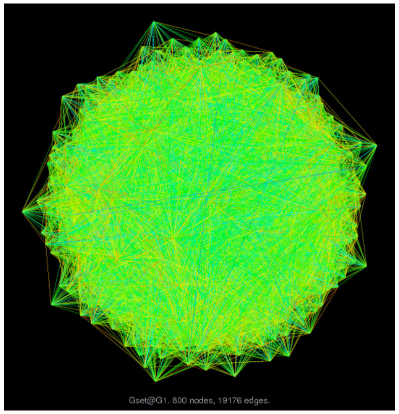
Leskovec, et al. (arXiv 2009); Mahdian-Xu 2007

- Small and large networks are very different:

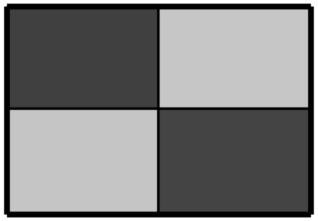
(also, an expander)

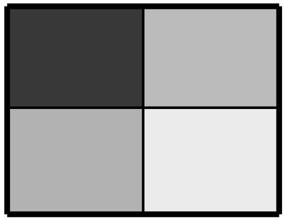


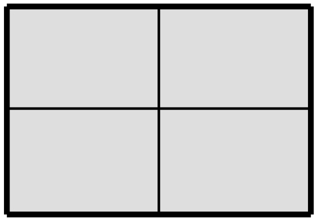




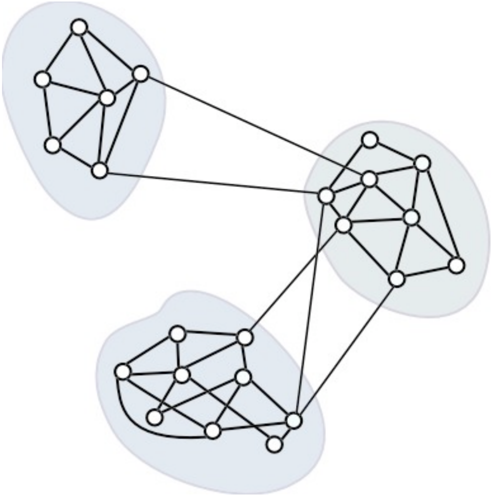
E.g., fit these networks to Stochastic Kronecker Graph with “base” K=[a b; b c]:

$K_1 =$ 





Implications: for Community Detection



- Linear (Low-rank) methods

  If Gaussian, then low-rank space is good.

- Kernel (non-linear) methods

  If low-dimensional manifold, then kernels are good

- Hierarchical methods

  Top-down and bottom-up -- common in the social sciences

- Graph partitioning methods

  Define "edge counting" metric -- conductance, expansion, modularity, etc. -- in interaction graph, then optimize!

(**Good** *and* **large**) network communities, *at least when formalized i.t.o. this bicriterion*, don't really exist in these graphs!!

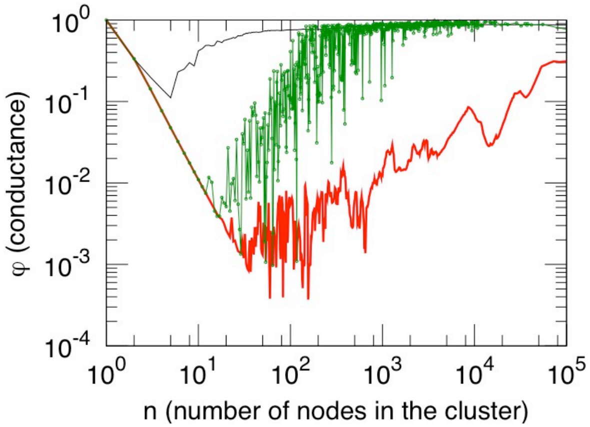
"It is a matter of common experience that communities exist in networks ... Although not precisely defined, communities are usually thought of as sets of nodes with better connections amongst its members than with the rest of the world."

Comparison with “Ground truth” (1 of 2)

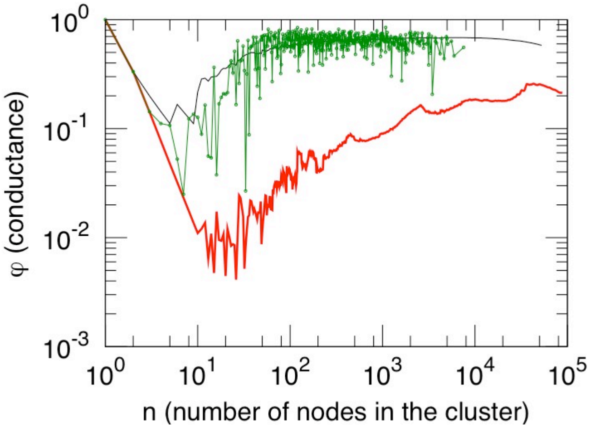
Networks with “ground truth” communities:

- LiveJournal12:
  - users create and explicitly join on-line groups
- CA-DBLP:
  - publication venues can be viewed as communities
- AmazonAllProd:
  - each item belongs to one or more hierarchically organized categories, as defined by Amazon
- AtM-IMDB:
  - countries of production and languages may be viewed as communities (thus every movie belongs to exactly one community and actors belongs to all communities to which movies in which they appeared belong)

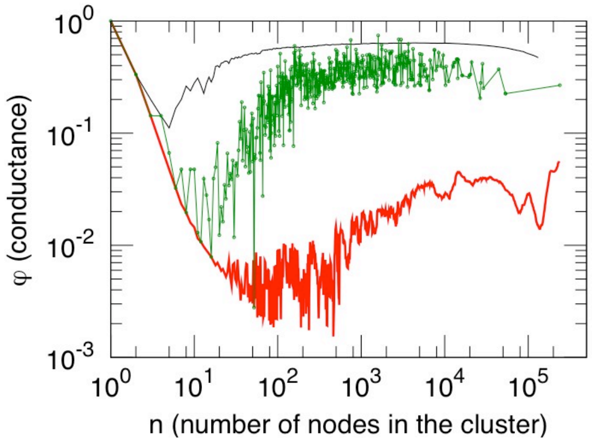
Comparison with "Ground truth" (2 of 2)



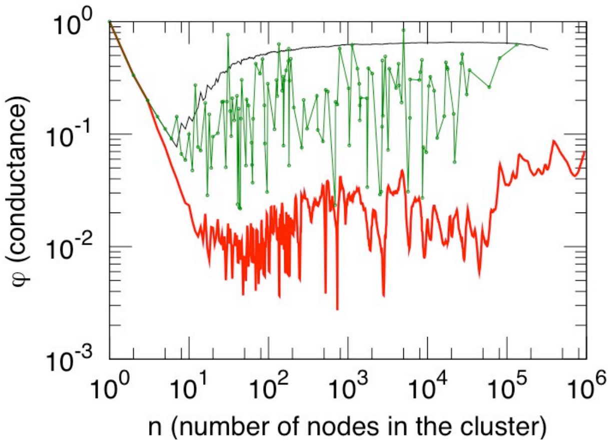
LiveJournal



CA-DBLP



AmazonAllProd



AtM-IMDB

Miscellaneous thoughts ...

Sociological work on community size (Dunbar and Allen)

- 150 individuals is maximum community size
- Military companies, on-line communities, divisions of corporations all ≤ 150

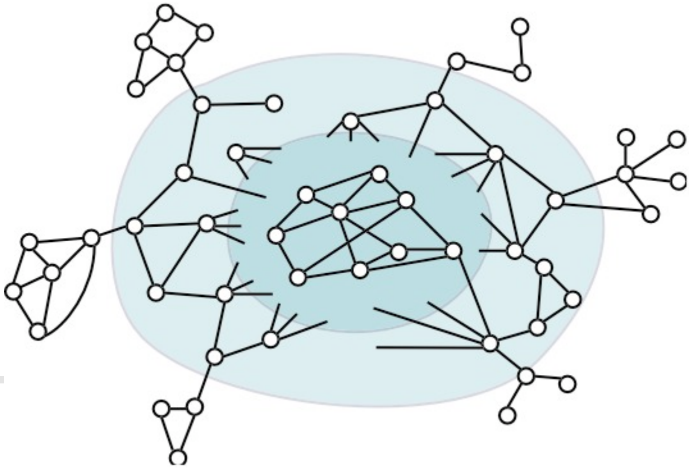
Common bond vs. common identity theory

- Common bond - people are attached to individual community members
- Common identity - people are attached to the group as a whole

What edges "mean" and community identification

- social networks - reasons an individual adds a link to a friend very diverse
- citation networks - links are more "expensive" and semantically uniform.

Implications: high level



What is simplest explanation for empirical facts?

- *Extremely* sparse Erdos-Renyi reproduces qualitative NCP (i.e., deep cuts at small size scales and no deep cuts at large size scales) since:

  sparsity + randomness = measure fails to concentrate

- Power law random graphs also reproduces qualitative NCP for analogous reason

- Iterative forest-fire model gives mechanism to put local geometry on sparse quasi-random scaffolding to get qualitative property of relatively gradual increase of NCP

Data are local-structure on global-noise, not small noise on global structure!

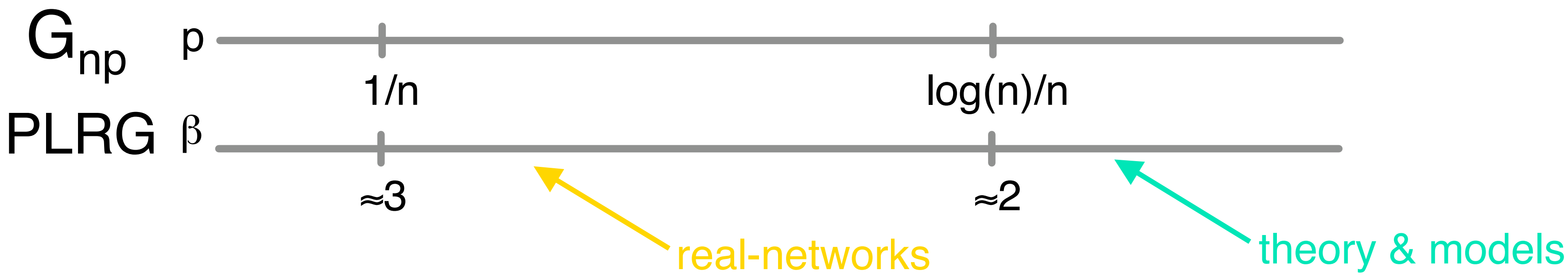
Implications: high level, cont.

Remember the Stochastic Kronecker theorem:

- Connected, if b+c>1: 0.55+0.15 > 1. No!
- Giant component, if (a+b)_(b+c)>1: (0.99+0.55)_(0.55+0.15) > 1. Yes!

Real graphs are in a region of parameter space analogous to *extremely sparse* G_{np}.

- Large vs small cuts, degree variability, eigenvector localization, etc.



G_{np} p

1/n log(n)/n

PLRG β

≈3 ≈2

real-networks

theory & models

Data are local-structure on global-noise, not small noise on global structure!

Degree heterogeneity and hyperbolicity

Social and information networks are expander-like at large size scales, but:

- Degree heterogeneity enhances hyperbolicity

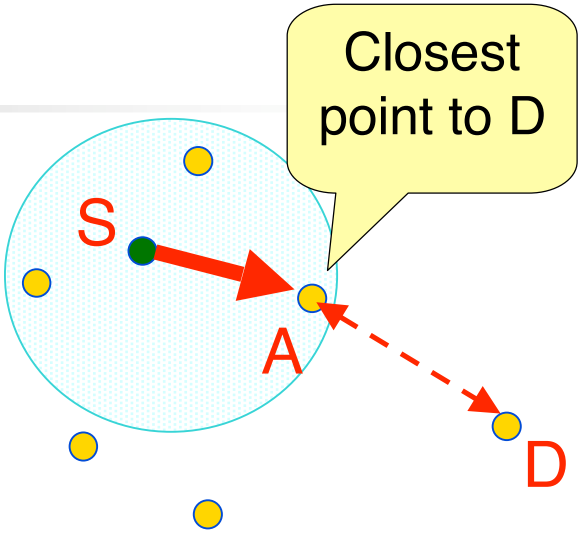
Lots of evidence:

- Scale free and internet graphs are more hyperbolic than other models, MC simulation - Jonckheere and Lohsoonthorne (2007)
- Mapping network nodes to spaces of negative curvature leads to scale-free structure - Krioukov et al (2008)
- Measurements of Internet are Gromov negatively curved - Baryshnikov (2002)
- Curvature of co-links interpreted as thematic layers in WWW - Eckmann and Moses (2002)

Question: Has anyone made this observation precise?

Hyperbolic Application 1: Internet Routing

A LARGE area - lots of other work.



Geographic routing protocols:

- A node knows (1) its location (physical or virtual coordinates), (2) its neighbors and their location, and (3) destination's location

- Forward packets to make progress to destination.

Euclidean versus Hyperbolic embeddings:

- Use virtual coordinates (Rao et al 2004, Fonseca et al 2005)

- Hyperbolic embeddings of same dimension do better (Shavitt and Tankel (2004,2008)

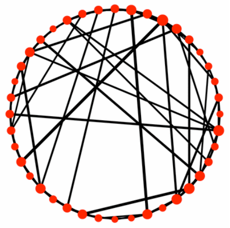
- Q: Which graphs have greedy embedding in the plane? (Papadimitriou and Rataczyk 2004)

- A: Every finite graph has greedy embedding in the hyperbolic plane. (R.Kleinberg 2005)

Hyperbolic Application 2: Decentralized Search in Social Graphs

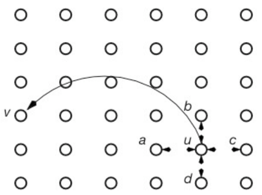
Milgram (1960s)

- Small world experiments - study short paths in social networks



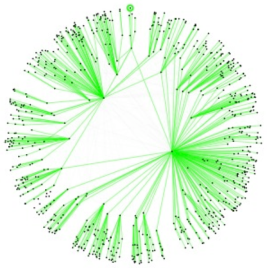
Watts and Strogatz (1998)

- Model that reproduce local clustering and existence of short paths



Kleinberg (2000)

- Model s.t. decentralized search can *find* short paths efficiently
- Careful coupling of "local" geometric structure and "global" structure.

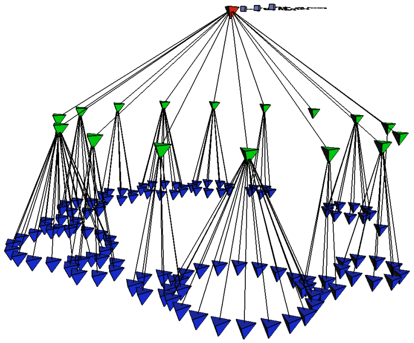


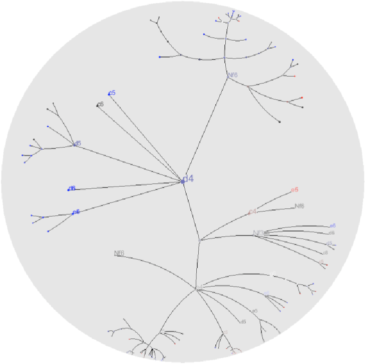
Boguna, Krioukov, and Claffy (2008)

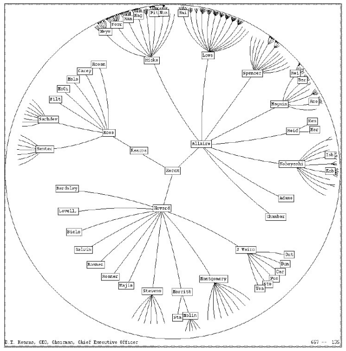
- Model with degree heterogeneity for efficient decentralized search
- *Analogous local-global coupling imply embedding in hyperbolic space*

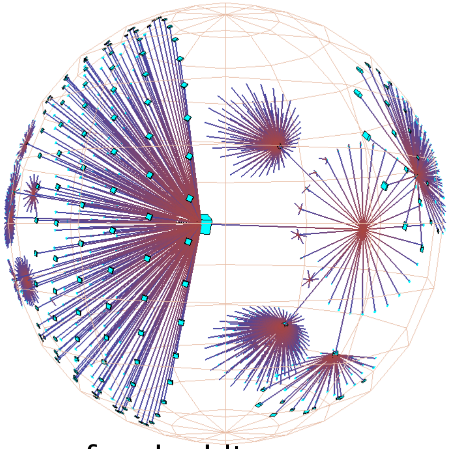
Hyperbolic Application 3: Internet and Web Visualization

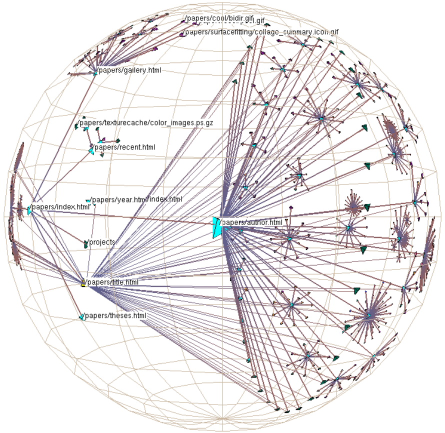
Munzner and Burchard (1995); Lamping, Rao, and Pirolli (1995); Munzner (1998)











Like the “fish-eye” camera lens, but avoids some ad-hoc decisions.

“There is no good way of embedding an exponentially growing tree in Euclidean space that allows us to simultaneously see both the entire structure and a closeup of a particular region. The solution is to use hyperbolic … geometry …” Munzner and Burchard (1995)

“Routing” versus “diffusion” metrics

Consider two classes of “distances” between nodes:

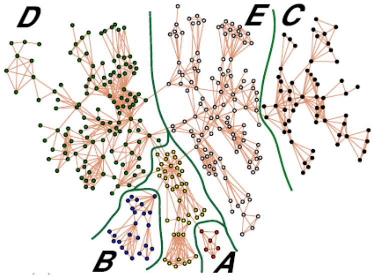
- “Diffusion-type” distance - related to (spectral methods and) diffusion or commute times

- “Geodesic-type” distance - related to (flow-based methods and) routing or shortest paths

Question 1: Which is better? More useful? (As a function of the type of graph)?

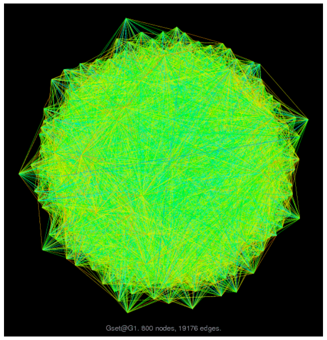
Question 2: Given that a process goes from A to B with one of those processes, how does the path compare with the other process?

Routing versus diffusions, cont*.



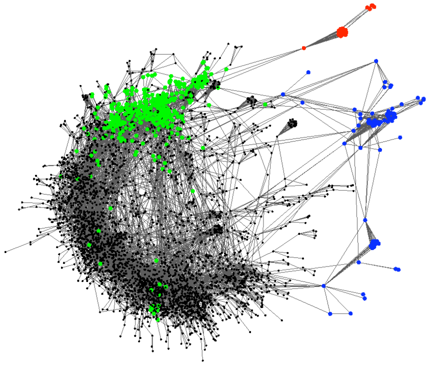
Low Dimensional Graphs

- Diffusions are discriminative and useful
- Flows and geodesics are too sensitive



Expander-like Graphs

- Diffusions not discriminitive or useful
- Multicommodity flow and geodesics useful?



Hyperbolic Graphs

- Diffusion path and routing path are the same.

*Question: Does anyone know of a formalization of this intuition?

Hyperbolic Application 4: Clustering and Community Structure

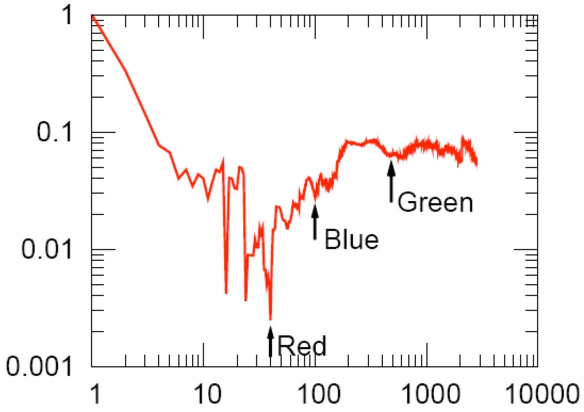
Hyperbolic properties at large size scales:

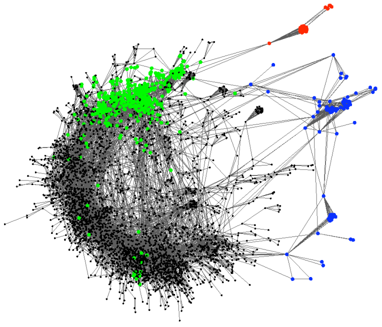
- (Degree-weighted) expansion at large size-scales

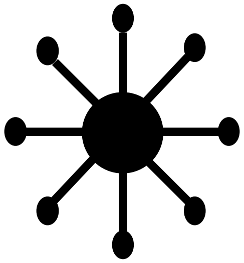
- Degree heterogeneity

Local pockets of structure on hyperbolic scaffolding.

- (Traditionally-conceptualized) communities get worse and worse as they get larger and larger







$$\begin{array}{|c|c|}\hline \alpha & \beta \\ \hline \beta & \gamma \\ \hline \end{array} = \begin{array}{|c|c|}\hline 0.99 & 0.55 \\ \hline 0.55 & 0.15 \\ \hline \end{array}$$

Implications for Data Analysis and ML

Principled and scalable algorithmic exploratory analysis tools:

- spectral vs. flow vs. combinations; local vs. global vs. improvement; etc.

Doing inference directly on data graphs, and machine learning in complex data environments:

- don't do inference on feature vectors with hyperplanes in a vector space
- need methods to do it in high-variability, *only approximately* low-dimensional, tree-like or expander-like environments.

Implicit regularization via approximate computation:

- spectral vs. flow vs. combinations; local vs. global vs. improvement; etc.

Data Application 1: Approximate eigenvector computation

Many uses of Linear Algebra in ML and Data Analysis involve *approximate* computations

- Power Method, Truncated Power Method, HeatKernel, Truncated Random Walk, PageRank, Truncated PageRank, Diffusion Kernels, TrustRank, etc.

- Often they come with a "generative story," e.g., random web surfer, teleportation preferences, drunk walkers, etc.

What are these procedures *actually* computing?

- E.g., what optimization problem is 3 steps of Power Method solving?

- Important to know if we really want to "scale up"

Implicit Regularization

Regularization: *A general method for computing "smoother" or "nicer" or "more regular" solutions - useful for inference, etc.*

Recall: Regularization is usually *implemented* by adding "regularization penalty" and optimizing the new objective.

$$\hat{x} = \mathrm{argmin}_x f(x) + \lambda g(x)$$

Empirical Observation: Heuristics, e.g., binning, early-stopping, etc. often implicitly perform regularization.

Question: Can approximate computation* *implicitly* lead to more regular solutions? If so, can we exploit this algorithmically?

*Here, consider approximate eigenvector computation. But, can it be done with graph algorithms?

Two versions of spectral partitioning

VP: ⟷ **SDP:**

$$\begin{array}{ll} \text{min.} & x^T L_G x \\ \text{s.t.} & x^T L_{K_n} x = 1 \\ & < x, 1 >_D = 0 \end{array}$$

$$\begin{array}{ll} \text{min.} & L_G \circ X \\ \text{s.t.} & L_{K_n} \circ X = 1 \\ & X \succeq 0 \end{array}$$

↓ ↓

R-VP:

$$\begin{array}{ll} \text{min.} & x^T L_G x + \lambda f(x) \\ \text{s.t.} & constraints \end{array}$$

R-SDP:

$$\begin{array}{ll} \text{min.} & L_G \circ X + \lambda F(X) \\ \text{s.t.} & constraints \end{array}$$

Views of approximate spectral methods

Three common procedures (L=Laplacian, and M=r.w. matrix):

- Heat Kernel: $H_t = \exp(-tL) = \sum_{k=0}^{\infty} \frac{(-t)^k}{k!} L^k$
- PageRank: $\pi(\gamma, s) = \gamma s + (1-\gamma) M \pi(\gamma, s)$

$$R_\gamma = \gamma \left(I - (1-\gamma) M \right)^{-1}$$

- q-step Lazy Random Walk: $W_\alpha^q = (\alpha I + (1-\alpha) M)^q$

Ques: Do these "*approximation* procedures" *exactly* optimizing some regularized objective?

A simple theorem

$$\text{(F,}\eta\text{)-SDP} \quad \begin{array}{ll} \min & L \bullet X + \frac{1}{\eta} \cdot F(X) \\ \text{s.t.} & I \bullet X = 1 \\ & X \succeq 0 \end{array}$$

Modification of the usual SDP form of spectral to have regularization (but, on the matrix X, not the vector x).

Theorem: Let G be a connected, weighted, undirected graph, with normalized Laplacian L. Then, the following conditions are sufficient for $X^\star$ to be an optimal solution to (F,η)-SDP.

- $X^\star = (\nabla F)^{-1} (\eta \cdot (\lambda^* I - L))$, for some $\lambda^* \in R$,
- $I \bullet X^\star = 1$,
- $X^\star \succeq 0$.

Three simple corollaries

$F_H(X) = \mathrm{Tr}(X \log X) - \mathrm{Tr}(X)$ (i.e., generalized entropy)

gives scaled Heat Kernel matrix, with $t = \eta$

$F_D(X) = -\mathrm{logdet}(X)$ (i.e., Log-determinant)

gives scaled PageRank matrix, with $t \sim \eta$

$F_p(X) = (1/p)||X||_p^p$ (i.e., matrix p-norm, for p>1)

gives Truncated Lazy Random Walk, with $\lambda \sim \eta$

These "approximation procedures" compute regularized versions of the Fiedler vector!

Large-scale applications

A lot of work on large-scale data already *implicitly* uses these ideas:

- Fuxman, Tsaparas, Achan, and Agrawal (2008): random walks on query-click for automatic keyword generation
- Najork, Gallapudi, and Panigraphy (2009): carefully "whittling down" neighborhood graph makes SALSA faster and better
- Lu, Tsaparas, Ntoulas, and Polanyi (2010): test which page-rank-like implicit regularization models are most consistent with data

Question: *Can we formalize this to understand when it succeeds and when it fails?*

Data Application 2: Classification in ML

Supervised binary classification

- Observe (X,Y) ε *(X,Y)* = (R^n , {-1,+1}) sampled from unknown distribution *P*
- Construct classifier α:*X->Y* (drawn from some family Λ, e.g., hyper-planes) after seeing *k* samples from unknown *P*

Question: How big must *k* be to get good prediction, i.e., low error?

- Risk: $R(\alpha)$ = probability that α misclassifies a random data point
- Empirical Risk: $R_{emp}(\alpha)$ = risk on observed data

Ways to bound | $R(\alpha) - R_{emp}(\alpha)$ | over all $\alpha \; \varepsilon \; \Lambda$

- VC dimension: distribution-independent; typical method
- Annealed entropy: distribution-dependent; but can get much finer bounds

Unfortunately …

Sample complexity of dstbn-free learning typically depends on the ambient dimension to which the data to be classified belongs

- E.g., Ω(d) for learning half-spaces in R^d.

Very unsatisfactory for *formally* high-dimensional data

- *approximately low-dimensional environments* (e.g., close to manifolds, empirical signatures of low-dimensionality, etc.)

- *high-variability environments* (e.g., heavy-tailed data, sparse data, pre-asymptotic sampling regime, etc.)

Ques: Can distribution-dependent tools give improved learning bounds for data with more realistic sparsity and noise?

Annealed entropy

Definition (Annealed Entropy): Let $\mathcal{P}$ be a probability measure on $\mathcal{H}$. Given a set Λ of decision rules and a set of points $Z = \{z_1, \ldots, z_\ell\} \subset \mathcal{H}$, let $N^\Lambda(z_1, \ldots, z_\ell)$ be the number of ways of labeling $\{z_1, \ldots, z_\ell\}$ into positive and negative samples. Then,

$$H^\Lambda_{ann}(k) := \ln E_{\mathcal{P}^{\times k}} N^\Lambda(z_1, \ldots, z_k)$$

is the *annealed entropy* of the classifier Λ with respect to $\mathcal{P}$.

Theorem: Given the above notation, the inequality

$$\text{Prob}\left[\sup_{\alpha\in\Lambda} \frac{R(\alpha) - R_{emp}(\alpha, \ell)}{\sqrt{R(\alpha)}} > \epsilon\right] < 4\exp\left(\left(\frac{H^\Lambda_{ann}(2\ell)}{\ell} - \frac{\epsilon^2}{4}\right)\ell\right)$$

holds true, for any number of samples ℓ and for any error parameter ϵ.

“Toward” learning on informatics graphs

Dimension-independent sample complexity bounds for

- High-variability environments
    - probability that a feature is nonzero decays as power law
    - magnitude of feature values decays as a power law
- *Approximately* low-dimensional environments
    - when have bounds on the covering number in a metric space
    - when use diffusion-based spectral kernels

Bound H_{ann} to get exact or gap-tolerant classification

Note: “toward” since we still learning in a vector space, not *directly* on the graph

Eigenvector localization …

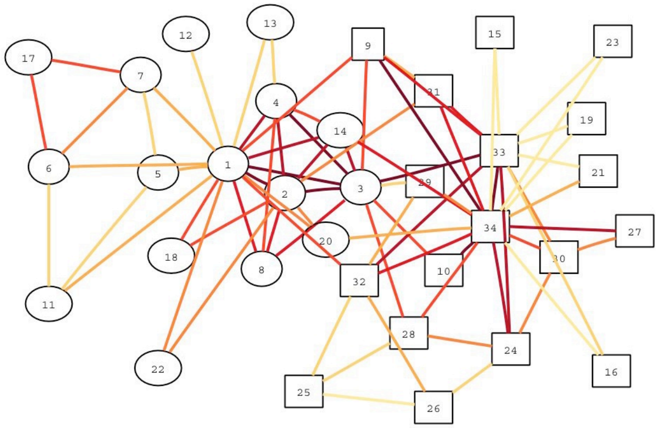
Let $\{f_i\}_{i=1}^{n}$ be the eigenfunctions of the normalized Laplacian of $\mathcal{L}_G$ and let $\{\lambda_i\}_{i=1}^{n}$ be the corresponding eigenvalues. Then, **Diffusion Maps** is:

$$\Phi : v \mapsto (\lambda_0^k f_0(v), \ldots, \lambda_n^k f_n(v)),$$

and **Laplacian Eigenmaps** is the special case of this feature map when $k = 0$.

When do eigenvectors localize?

- High degree nodes.
- Articulation/boundary points.
- Points that "stick out" a lot.
- Sparse random graphs



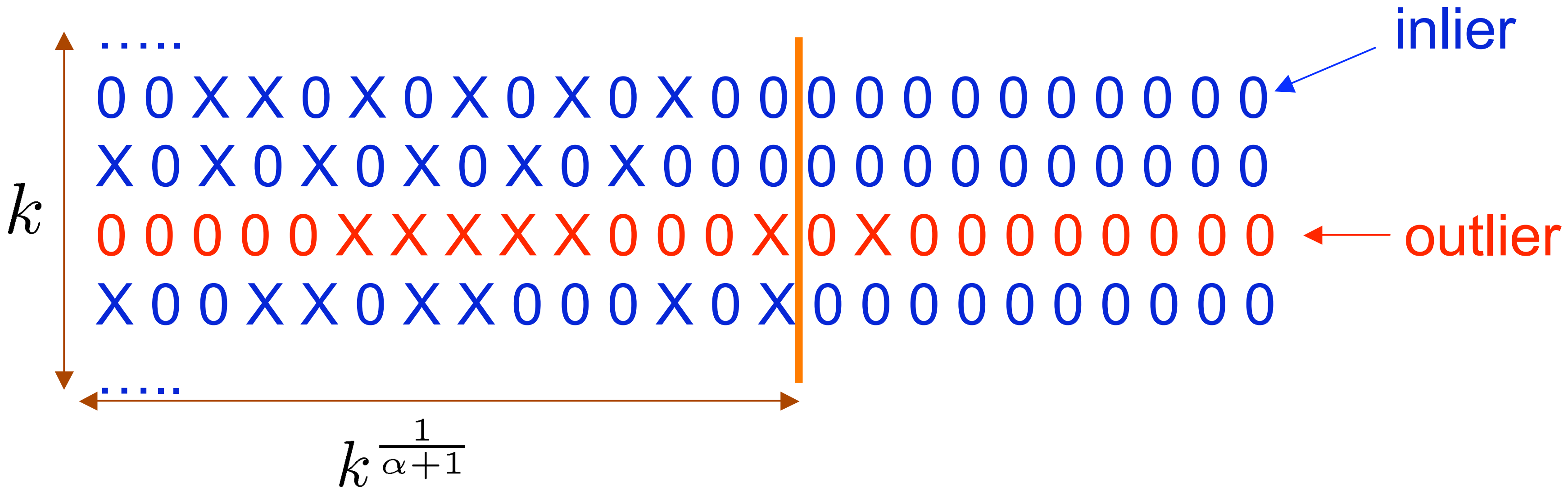
This is seen in many data sets when eigen-methods are chosen for algorithmic, and not statistical, reasons.

Exact learning with a heavy-tail model

Mahoney and Narayanan (2009,2010)

Heavy-tailed model: Let $\mathcal{P}$ be a probability distribution in R^d. Suppose $\mathcal{P}[x_i \neq 0] \leq Ci^{-\alpha}$ for some absolute constant $C > 0$, with $\alpha > 1$.

Theorem: In this model, $H^{\Lambda}_{ann}(\ell) \leq \left(\frac{C}{\alpha-1}\ell^{\frac{1}{\alpha}} + 1\right)\ln(\ell)$. Thus, need only $\ell = \tilde{O}\left(\left(\frac{C\ln(\delta^{-1})}{\epsilon^2}\right)^{\frac{\alpha+1}{\alpha}}\right)$ samples, independent of (possibly infinite) d.



.....

0 0 X X 0 X 0 X 0 X 0 X 0 0 | 0 0 0 0 0 0 0 0 0 0

X 0 X 0 X 0 X 0 X 0 X 0 0 0 | 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 X X X X X 0 0 0 X | 0 X 0 0 0 0 0 0 0 0

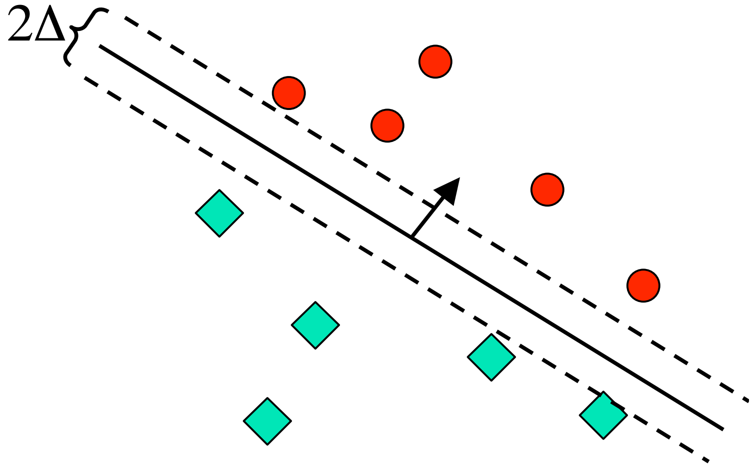
X 0 0 X X 0 X X 0 0 0 X 0 X | 0 0 0 0 0 0 0 0 0 0

.....

Gap-tolerant classification

Mahoney and Narayanan (2009,2010)

Def: A *gap-tolerant classifier* consists of an oriented hyper-plane and a margin of thickness Δ around it. Points outside the margin are labeled ±1; points inside the margin are simply declared "correct."



Only the expectation of the norm needs to be bounded! Particular elements can behave poorly!

Theorem: Let $\mathcal{P}$ be a probability measure on a Hilbert space $\mathcal{H}$, and let $\Delta > 0$. If $E_{\mathcal{P}}\|x\|^2 = r^2 < \infty$, then then the annealed entropy of gap-tolerant classifiers in $\mathcal{H}$, where the gap is Δ, is

$$H_{ann}^{\Delta}(\ell) \leq \left(\ell^{\frac{1}{2}}\left(\frac{r}{\Delta}\right)+1\right)(1+\ln(\ell+1)).$$

so can get dimension-independent bounds!

Large-margin classification with very "outlying" data points

Mahoney and Narayanan (2009,2010)

Apps to dimension-independent large-margin learning:

- with **spectral kernels**, e.g. Diffusion Maps kernel underlying manifold-based methods, on arbitrary graphs

- with **heavy-tailed data**, e.g., when the magnitude of the elements of the feature vector decay in a heavy-tailed manner

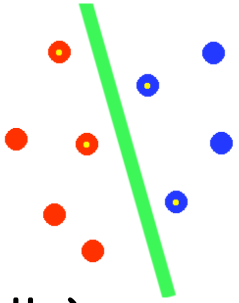
Technical notes:

- new proof bounding VC-dim of gap-tolerant classifiers in Hilbert space generalizes to Banach spaces - useful if dot products & kernels too limiting

- Ques: *Can we control aggregate effect of "outliers" in other data models?*

- Ques: *Can we learn if measure never concentrates?*

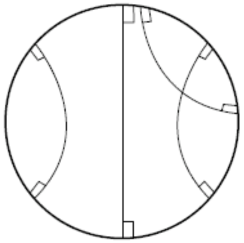
Data application 2, more generally …



Machine learning in environments more general than R^n or RKHS?

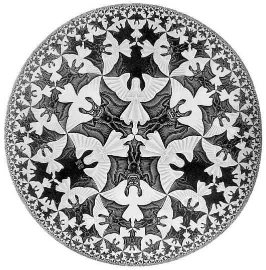
- On expander-like or hyperbolic structures (locally/globally)

- On other classes of metric spaces, while exploiting metric/geometric structure for learning?



- How do ideas like margin, etc. generalize?

Learn *directly* on graph (non-vector/matrix) data



- i.e., don't filter through vector space, but perform capacity control, etc directly on graph

- don't assume m,n,p -> Infty in a nice way

Conclusions (1 of 4): General Observations

Network data are often very/extremely large:

- Premium on fast/scalable algorithms
- (Good - lots of algorithms; Bad - they often return meaningless answers.)

Network data are often very/extremely sparse:

- Premium on statistical regularization
- (Good - lots of regularization methods; Bad - they work on vectors, not graphs.)
- BTW, this implies "landmark point methods" often inappropriate

Networks have complex, nonlinear, adversarial structure

- Structures don't exist in small (e.g.,thousands of nodes) networks
- Need tools to explore things we can't visualize
- Big difference between "analyst appls" and "next-user-interaction apps"

Conclusions (2 of 4): General Observations

- Algorithmic primitives to "probe" networks locally and globally
- Infer properties of original network from statistical and regularization properties of ensembles of approximate solutions
- Real informatics graphs -- very different than small commonly-studied graphs and existing generative models
- Tools promising for coupling local properties (often low-dimensional) and global properties (often expander-like)
- Tools promising to study pre-existing geometry versus generated geometry - recall *geometry* $\approx$ *inference*
- *Validation is difficult* - if you have a clean validation and/or a pretty picture, you're looking at unrealistic network data!

Conclusion (3 of 4) : "Structure" and "randomness" in large informatics graphs

High-level observations to formalize:

- There do not exist a "small" number of linear components that capture "most" of the variance/information in the data.
- There do not exist "nice" manifolds that describe the data well.
- There is "locally linear" structure or geometry on small size scales that does not propagate to global/large size scales.
- At large size scales, the "true" geometry is more "hyperbolic" or "tree-like" or "expander-like".

Important: *even if you do not care about communities, conductance, hyperbolicity, etc., these empirical facts place very severe constraints on the types of models and types of analysis tools that are appropriate.*

Conclusion (4 of 4): Geometric Network Analysis Tools?

- Approximation algorithms have geometry hidden somewhere

  Spectral methods, LP methods, tree methods, metric embeddings

- Local Spectral Methods

  Identify geometry at multiple nodes at multiple size scales

  No need to assume local geometries are on a global manifold

- Approximate Computation as Implicit Regularization

  Approximate solutions are better than exact solutions

  Especially relevant for extremely sparse/noisy networks

  Use this to regularize and do inference directly on network?

- Methodological test case

  Good "hydrogen atom" for development of algorithmic and statistical tools for probing graph data more generally