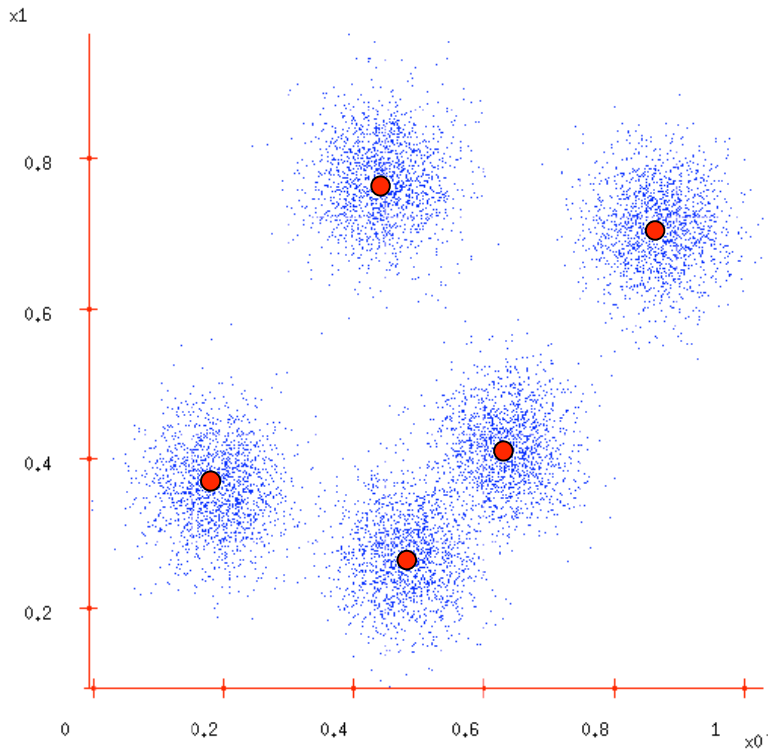
k-means clustering, cont'd



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:

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

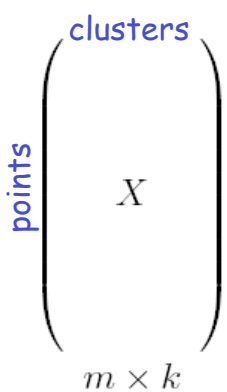
Note: The *intuition* underlying the combinatorial objective is that there are several "nice" clusters in a low-dimensional space.

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.



- 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., $X^T X = I$.

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

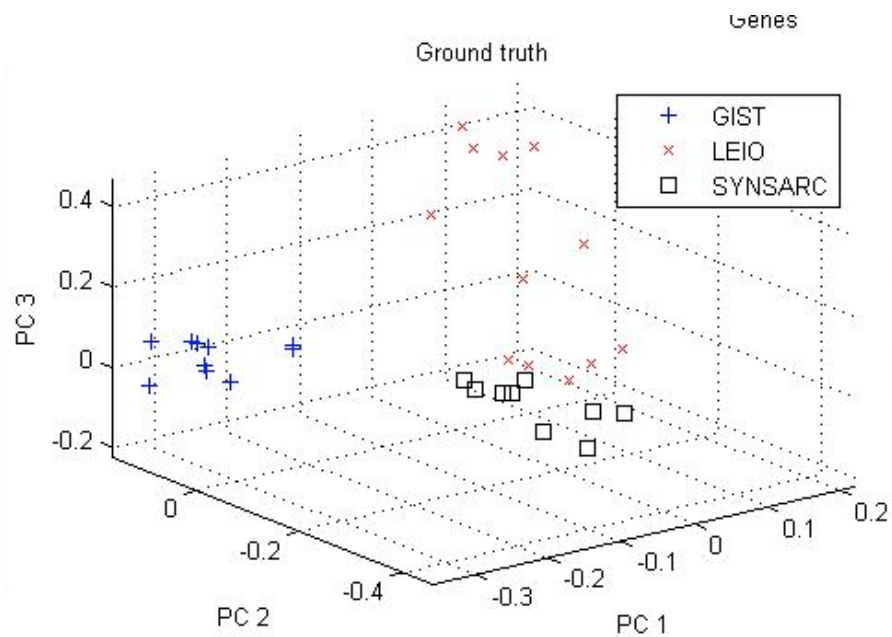
$$\min_{X \in \mathbb{R}^{m \times k}} \|A\|_F^2 - \left\|X^T A\right\|_F^2 \qquad \text{or} \qquad \max_{X \in \mathbb{R}^{m \times k}} \left\|X^T A\right\|_F^2$$

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

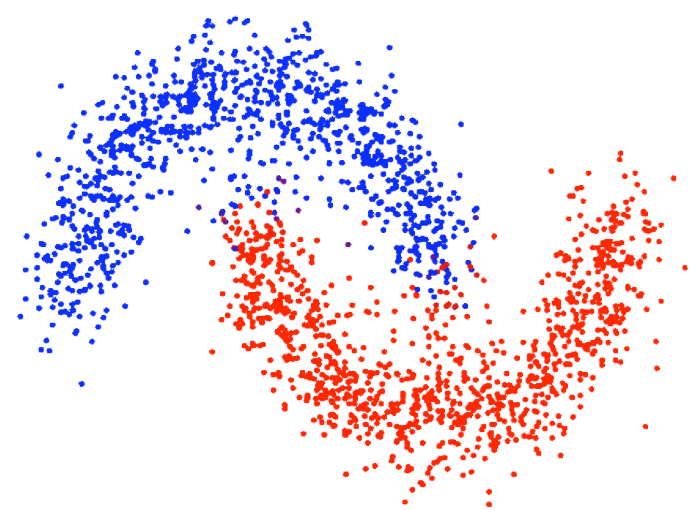
Using SVD to solve *k*-means

- 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.

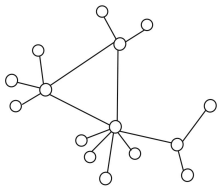
Eigenvectors identify linear structure

- 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?

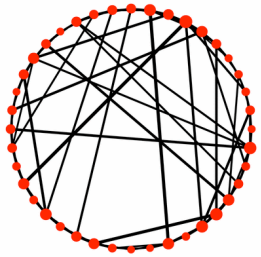
Unfortunately …



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

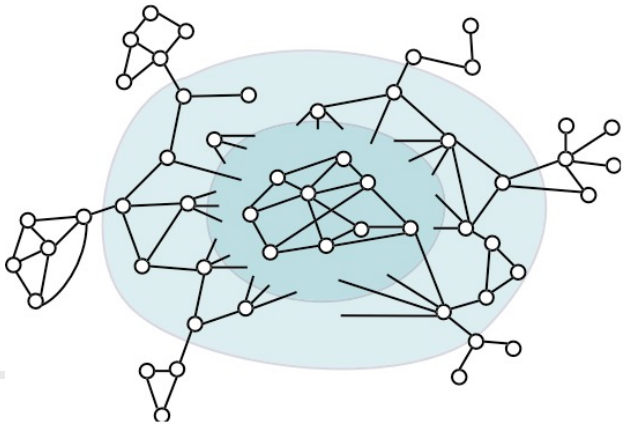
- Relationship governs diffusion of information; decentralized search; routing; dynamic properties; applicability of common ML tools

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

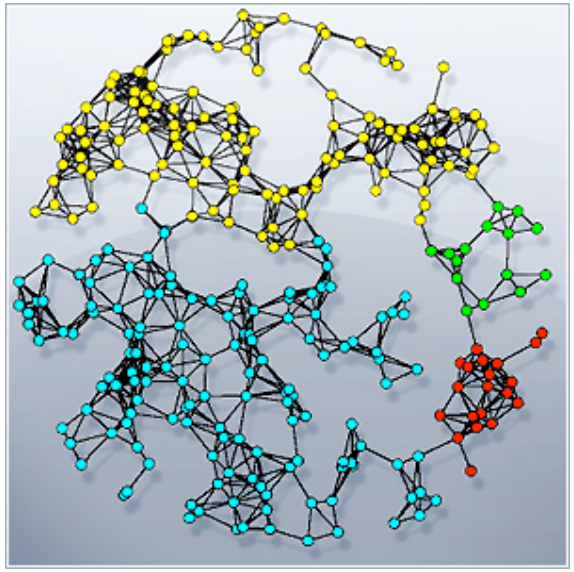
- Spectral, flow, multi-resolution algorithms; their implicit geometric basis; global and scalable local methods; expander-like, tree-like, and hyperbolic structure

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 …

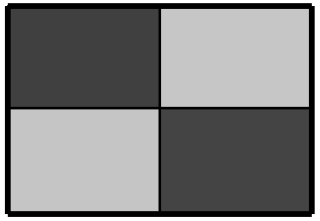
Say we want to find a "best fit" of the adjacency matrix to:

α	β
β	γ

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

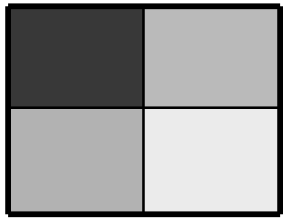
$\alpha \approx \gamma \gg \beta$

low-dimensional



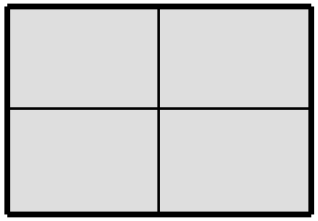
$\alpha \gg \beta \gg \gamma$

core-periphery



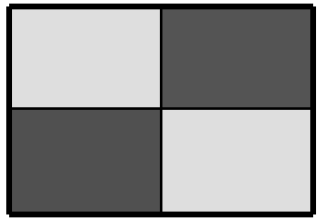
$\alpha \approx \beta \approx \gamma$

expander or K_n



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

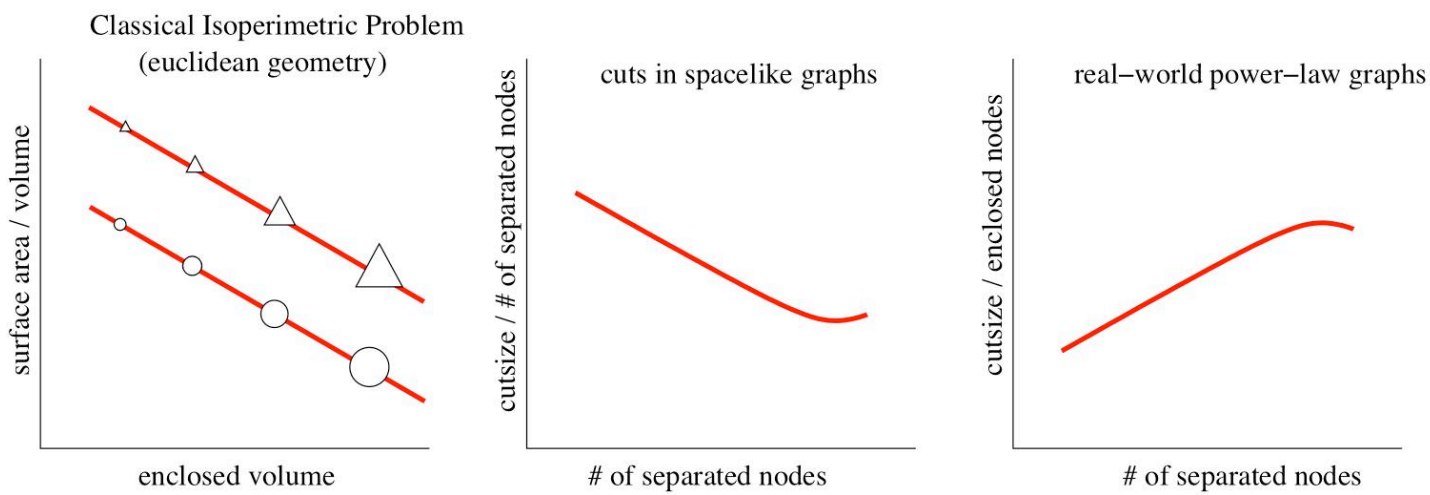
bipartite graph



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

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

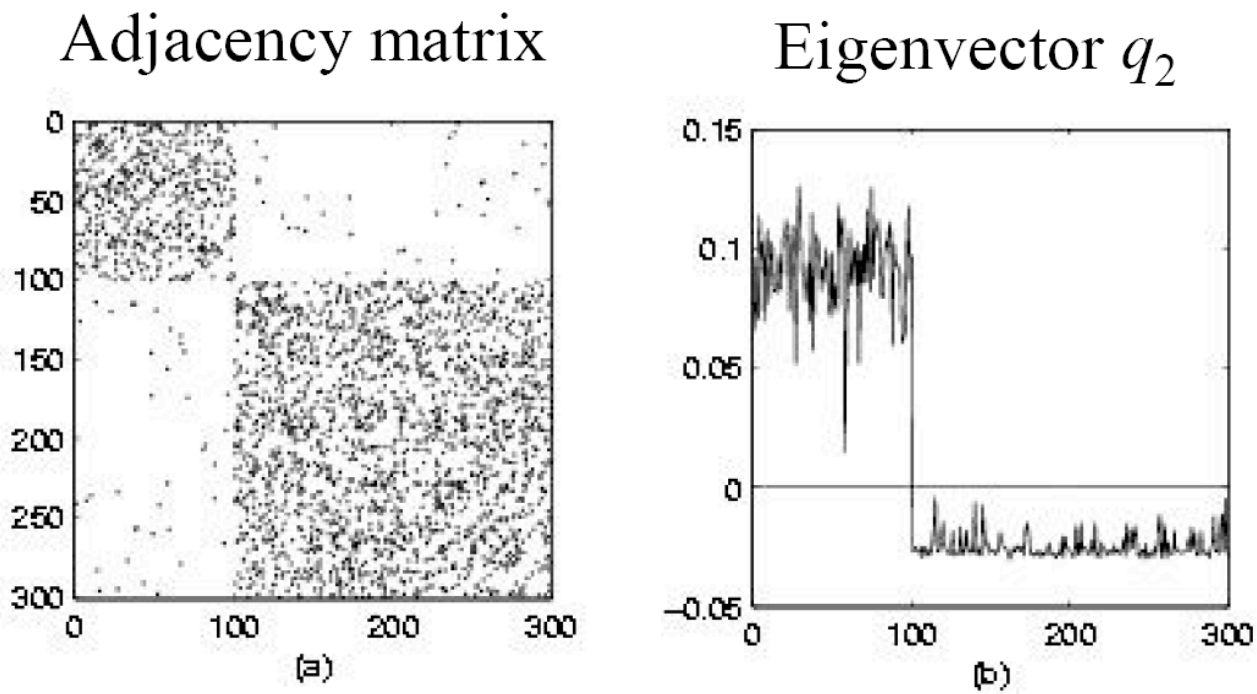
Note: only need to get an approximate eigenvector.

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:

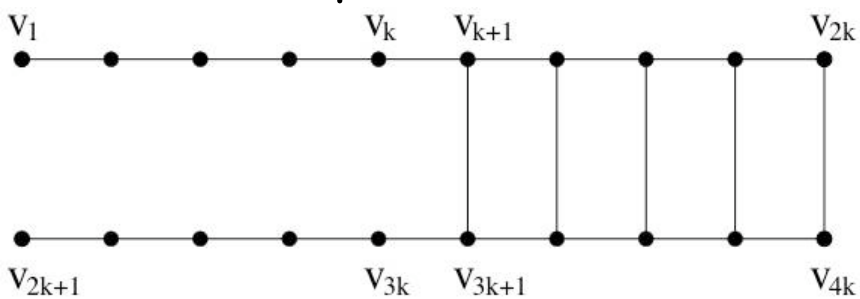


Note: "Looks" like k-means when cuts are well-balanced.

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 $x \perp D1$, this is equivalent to:

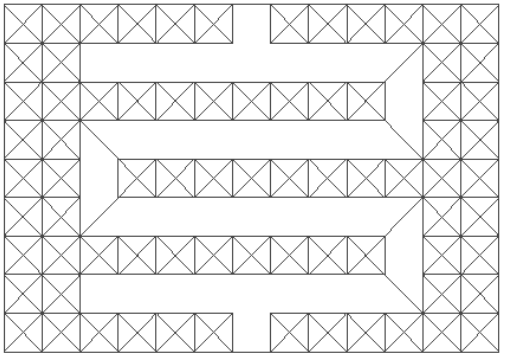
$$\frac{\lambda_1}{\mathrm{vol(G)}} = \min_{x \perp D1} \frac{\sum_{i \sim j} (x_i - x_j)^2}{\sum_{i,j} (x_i - x_j)^2 d_i d_j}$$

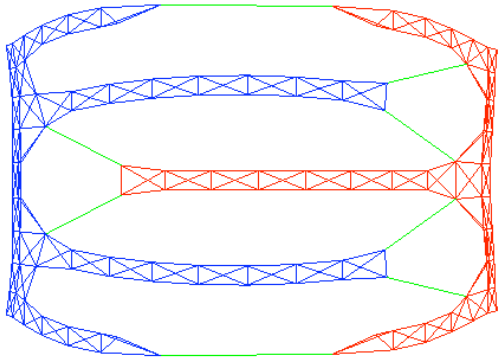
Interpretation:

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

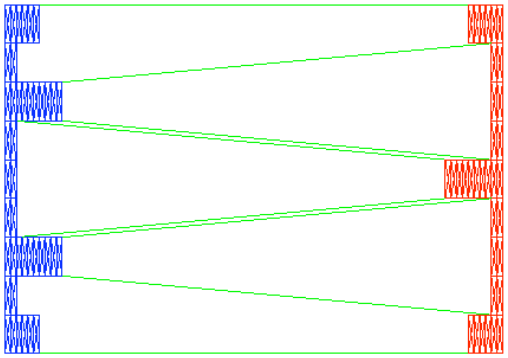
"Regularization" and spectral methods

- 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

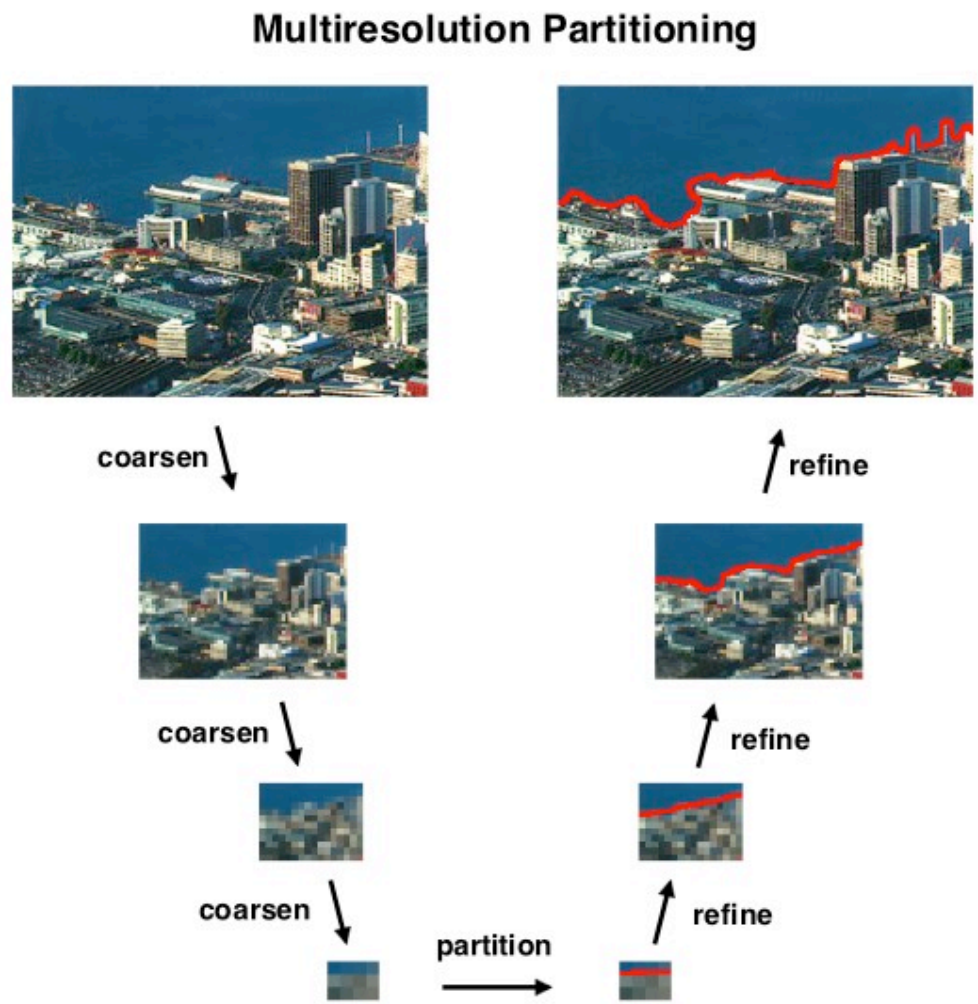
Chaco (1993)

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

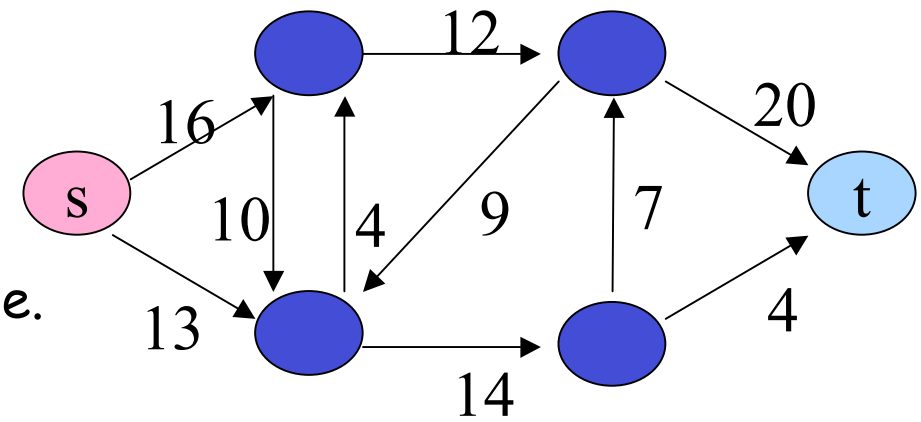
Metis (1995)

- 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 $s \in V$, sink $t \in V$.
- Capacity $c(e) \in \mathbf{Z}^+$ for each edge e.
- Flow: function $f: E \to \mathbf{N}$ s.t.
- For all e: $f(e) \leq c(e)$
- 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 $e \varepsilon E$, depending on whether edge e is cut

$y(i) = 0,1$, for $i \varepsilon k$ (commodities), depending if commodity i disconnected

P_i, $i \varepsilon k$, 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 $d: V \times V \to R+$.

(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 $T \subset V$, 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 $S \subseteq T$ 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

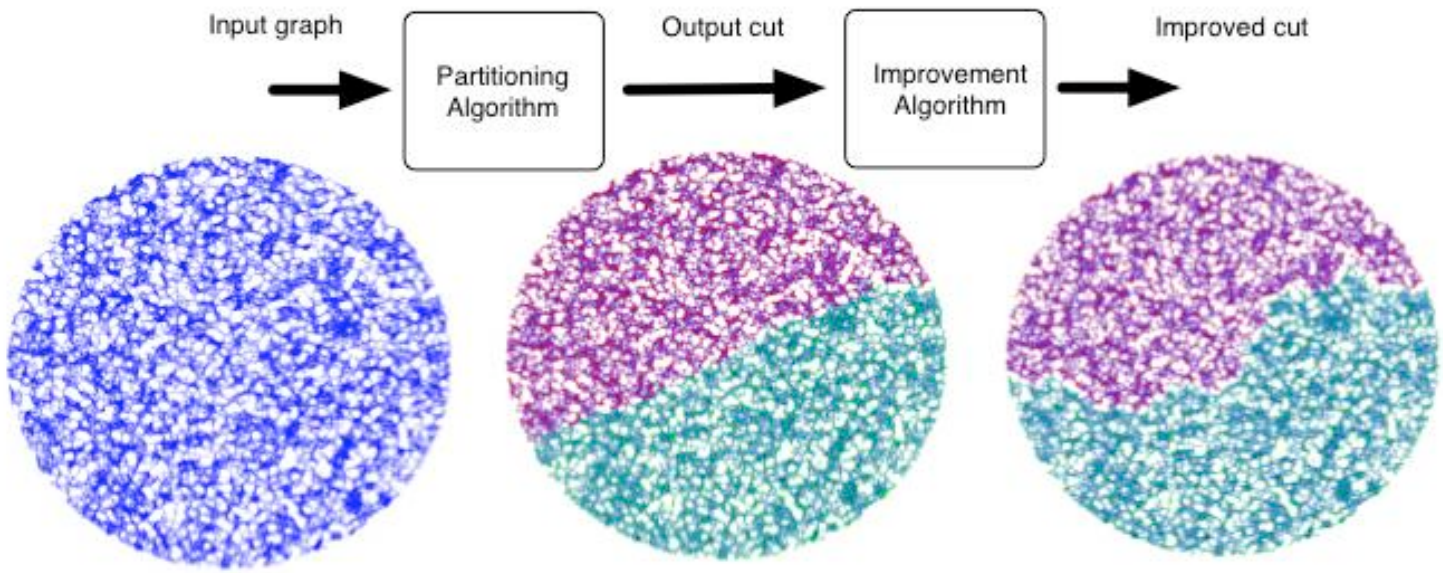
Uses of flow-based cut-improvement algorithms

- 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

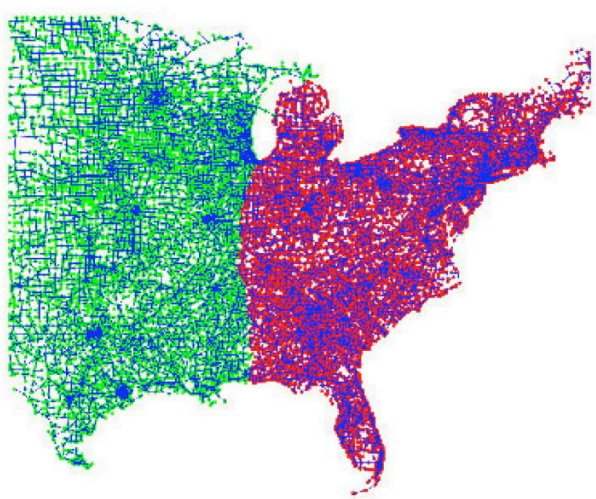
Local spectral methods

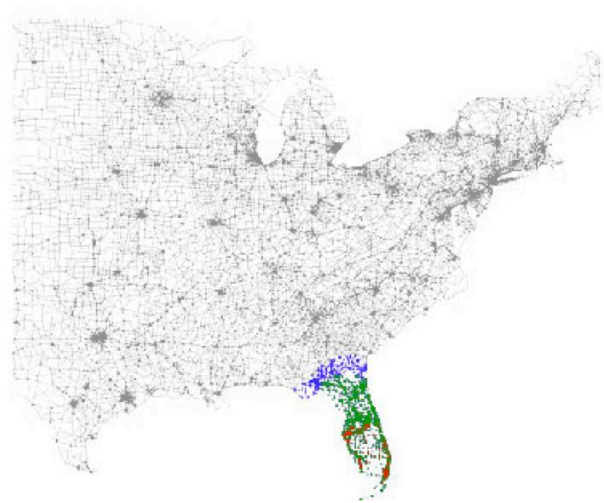
Local spectral methods - provably-good local version of global spectral

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:

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

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)

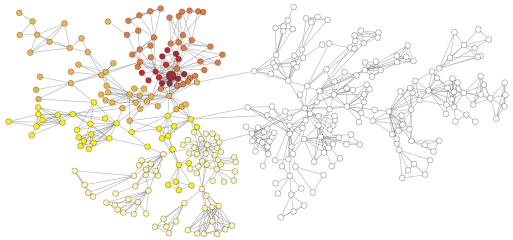
Mahoney, Orecchia, and Vishnoi (2010)

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

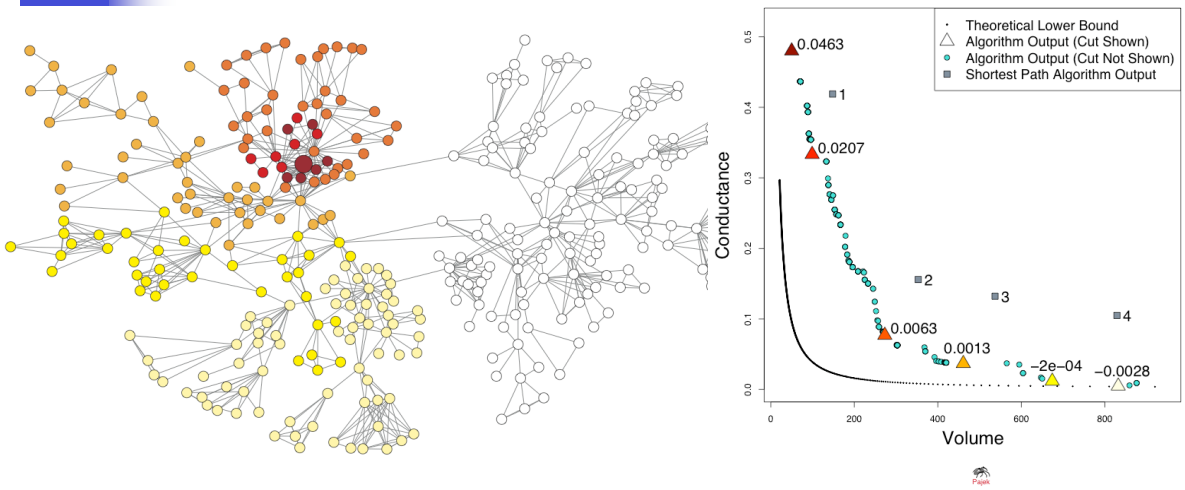
Flow improvement methods - Given a graph G and a partition, find a "nearby" cut that is of similar quality:

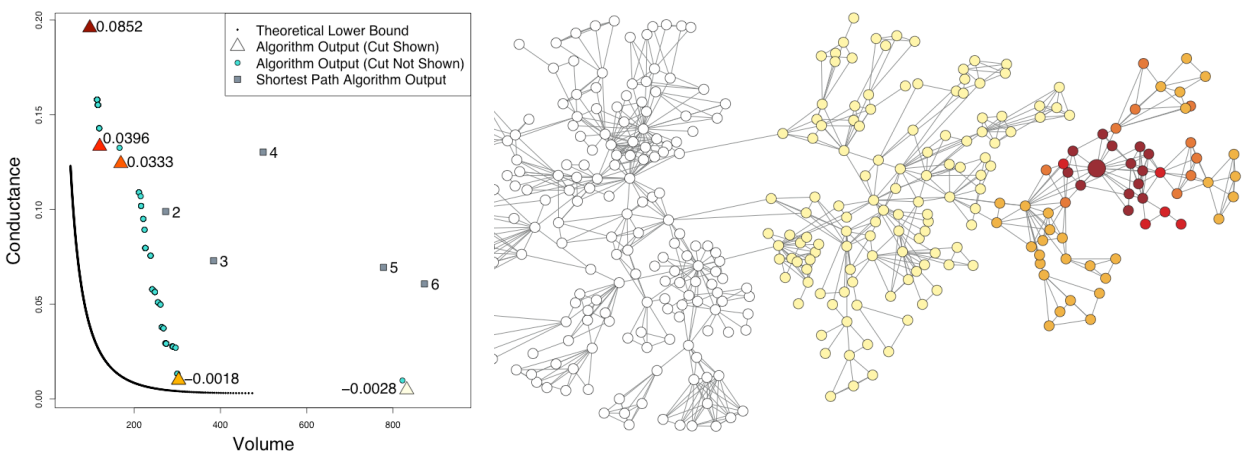
GGT89: find min conductance subset of a "small" partition

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

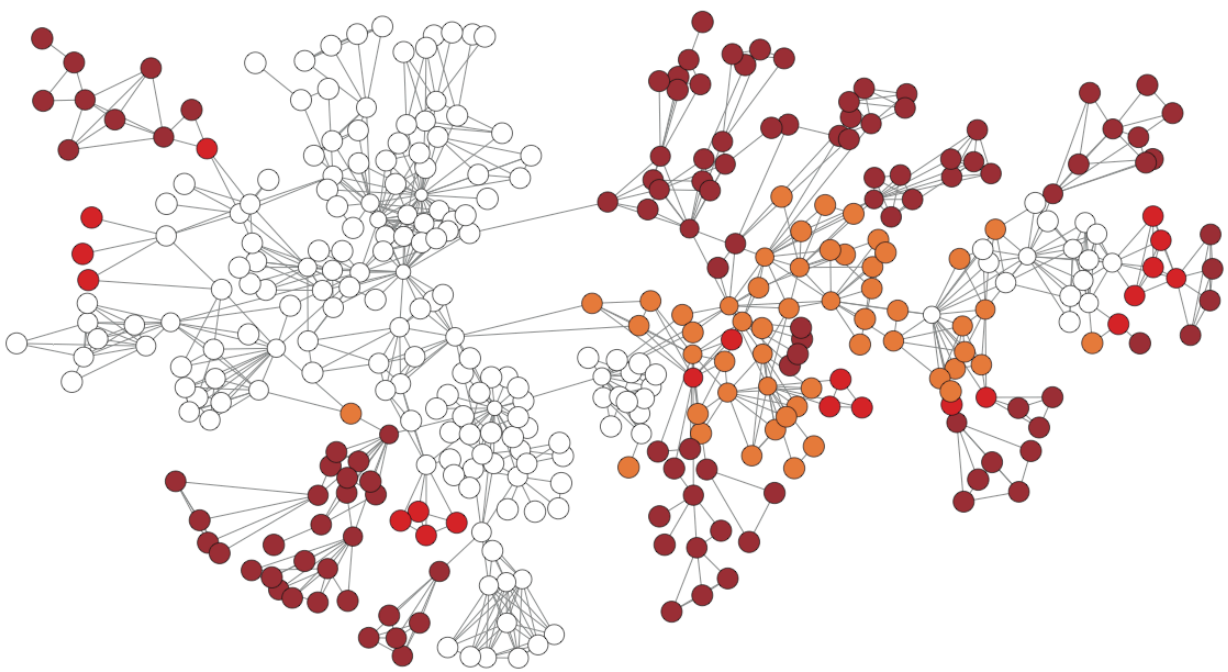




- Similar results if we do local random walks, truncated PageRank, and heat kernel diffusions.
- Often, it finds "worse" quality but "nicer" partitions than flow-improve methods. (Tradeoff we'll see later.)

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. $\mathrm{vol}(C) \leq \mathrm{vol}(\bar{C})$ and $\mathrm{vol}(T) \leq \mathrm{vol}(\bar{T})$:

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

Spectral and flow correlation measures are related:

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

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$\cap L2^2$ - 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.

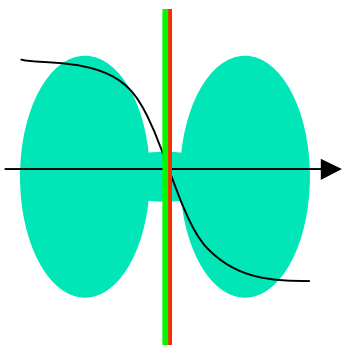
OSVV "spectral-flow" partitioning

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

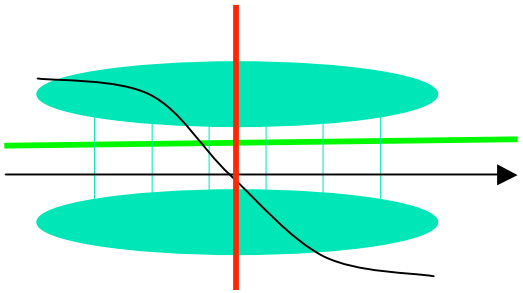
SPECTRAL

- 2nd eigenvector
- Spectral cut
- Optimal cut

GOOD CASE

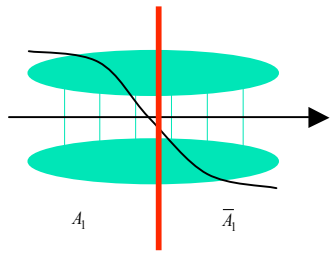


BAD CASE: LONG PATHS

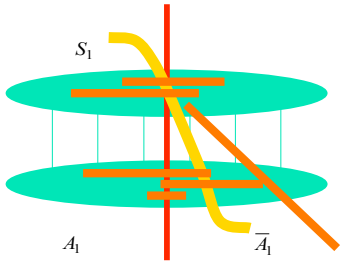


OSVV

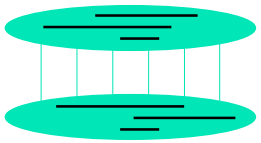
SPECTRAL STEP



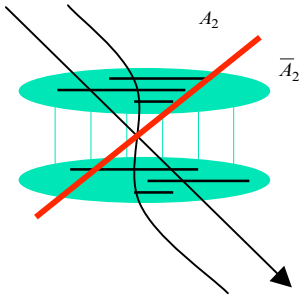
FLOW IMPROVEMENT STEP



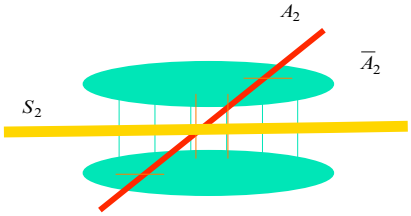
$G_1 = G + M_1$



SPECTRAL STEP



FLOW IMPROVEMENT STEP



OPTIMAL CUT FOUND