Normalized Cuts Without Eigenvectors: A Multilevel Approach

Inderjit S. Dhillon

The University of Texas at Austin

SIAM Conference on Parallel Processing

February 24, 2006

Joint work with Yuqiang Guan and Brian Kulis

Clustering

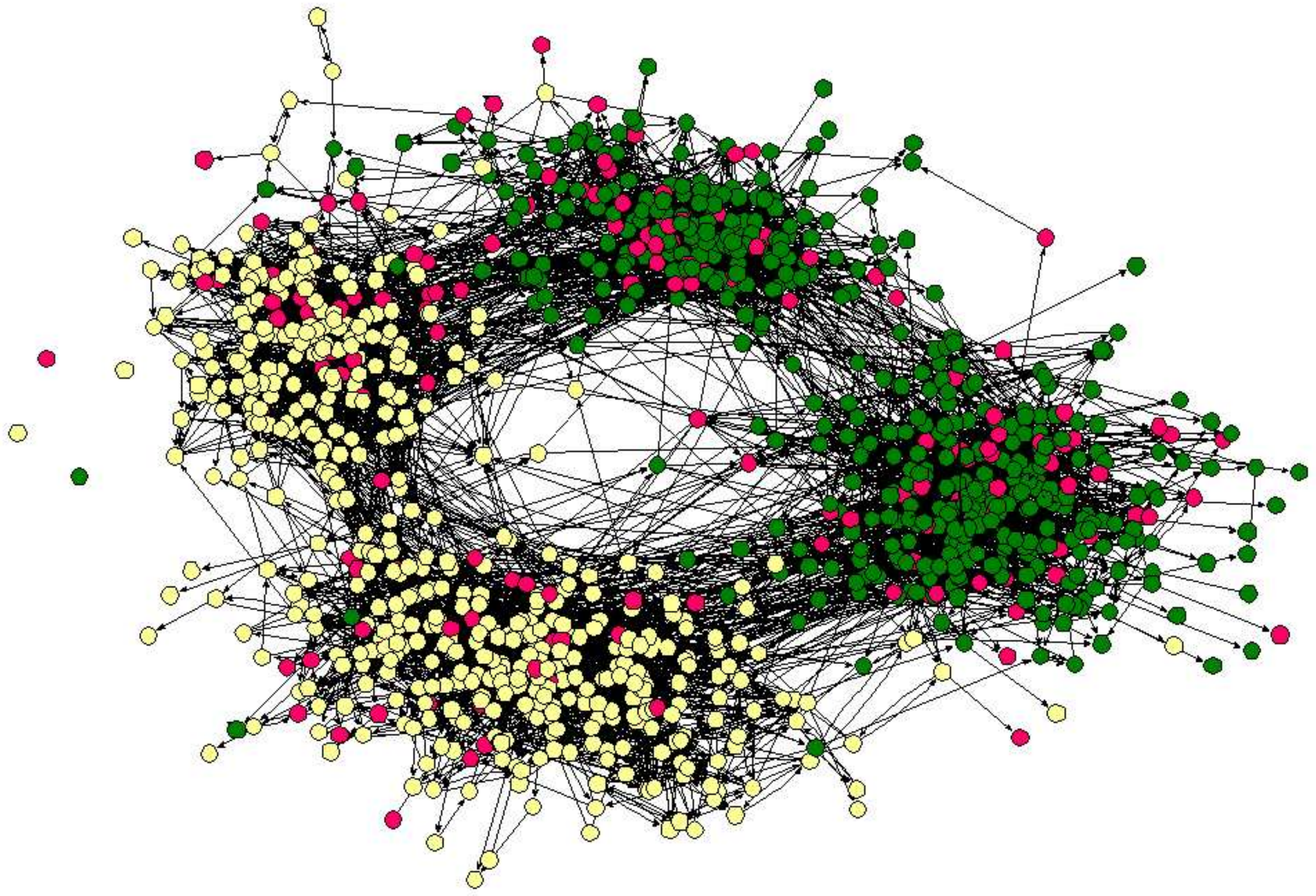
Partitioning data into clusters arises in various applications in data mining & machine learning.

Examples:

- Bioinformatics: Identifying similar genes
- Text Mining: Organizing document collections
- Image/Audio Analysis: Image and Speech segmentation
- Web Search: Clustering web search results
- Social Network Analysis: Identifying social groups
- Other: Load balancing and circuit partitioning

Graph Partitioning/Clustering

- In many applications, the goal is to partition/cluster the nodes of a graph:

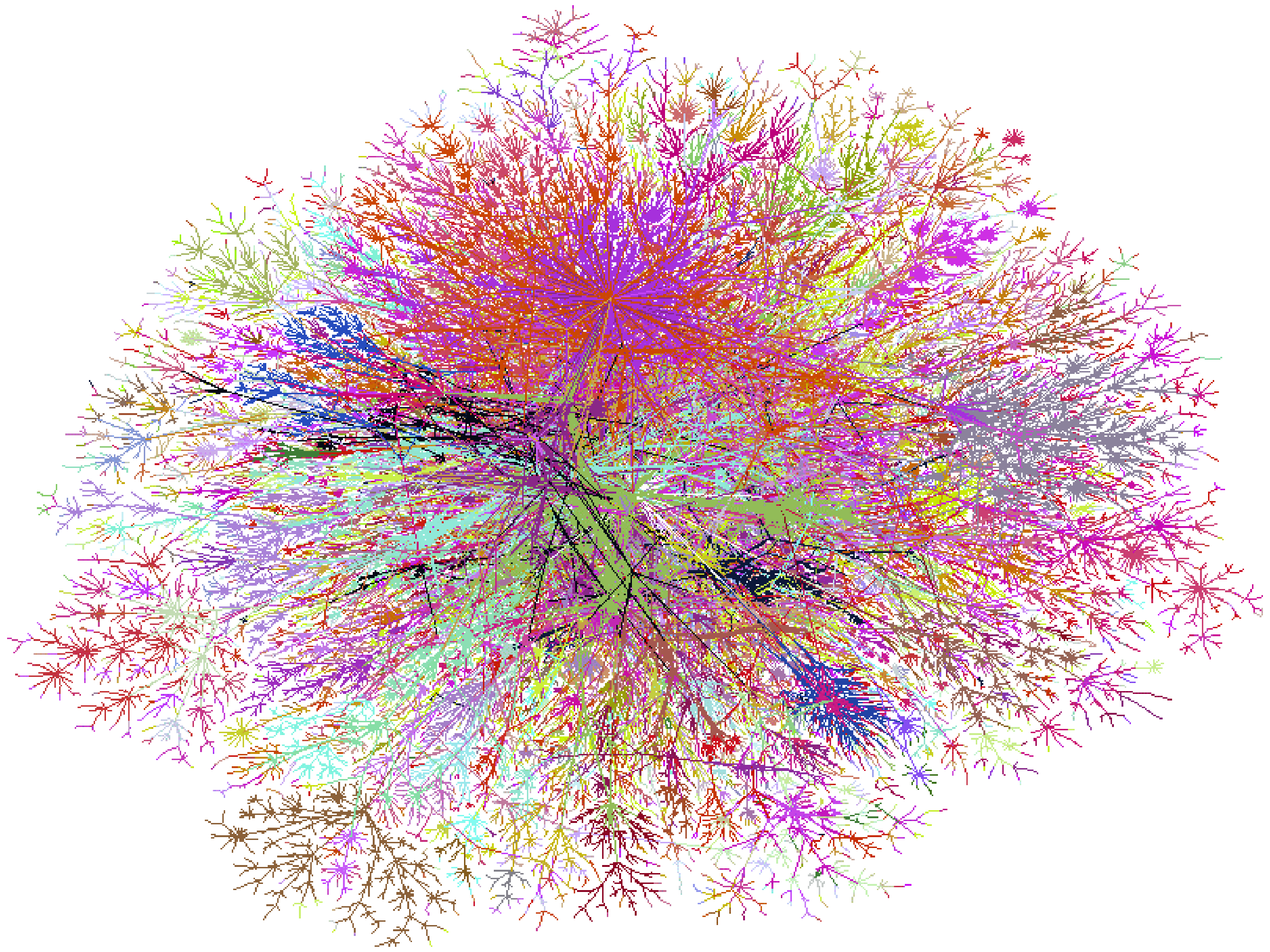


High School Friendship Network

[James Moody. American Journal of Sociology, 2001]

Graph Partitioning/Clustering

- In many applications, the goal is to partition/cluster the nodes of a graph:



The Internet

[The Internet Mapping Project, Hal Burch and Bill Cheswick, Lumeta Corp, 1999]

Graph Clustering Objectives

- How do we measure the quality of a graph clustering?

Graph Clustering Objectives

- How do we measure the quality of a graph clustering?
- Could simply minimize the *edge-cut* in the graph
    - Can lead to clusters that are highly unbalanced in size

Graph Clustering Objectives

- How do we measure the quality of a graph clustering?
- Could simply minimize the *edge-cut* in the graph
  - Can lead to clusters that are highly unbalanced in size
- Could minimize the *edge-cut* in the graph while constraining the clusters to be equal in size
  - Not a natural restriction in data analysis

Graph Clustering Objectives

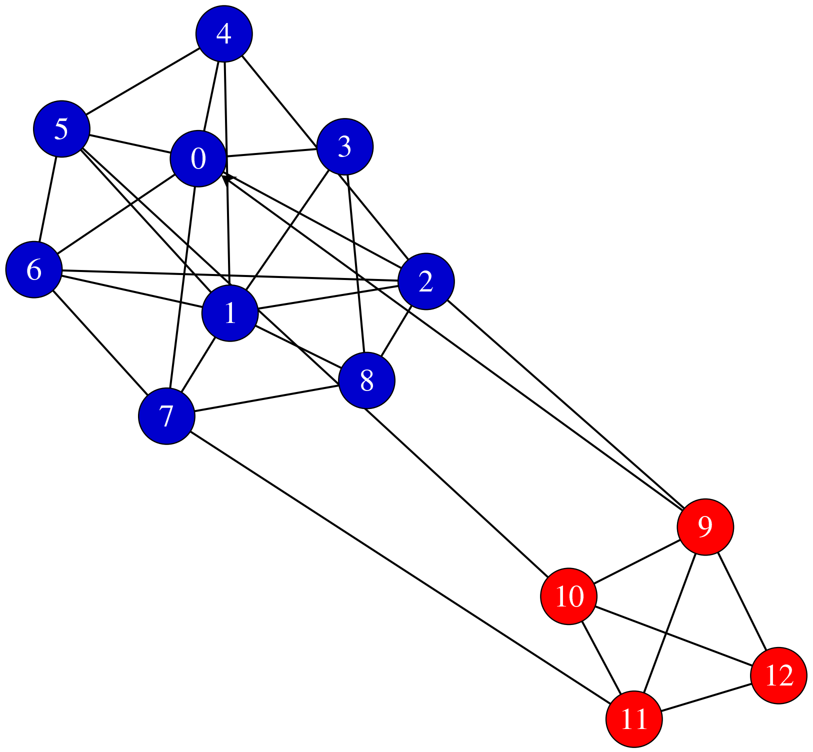
- How do we measure the quality of a graph clustering?
- Could simply minimize the *edge-cut* in the graph
  - Can lead to clusters that are highly unbalanced in size
- Could minimize the *edge-cut* in the graph while constraining the clusters to be equal in size
  - Not a natural restriction in data analysis
- Popular objectives include normalized cut, ratio cut and ratio association

Normalized Cut: minimize $$\sum_{i=1}^{c} \frac{\text{links}(\mathcal{V}_i, \mathcal{V} \setminus \mathcal{V}_i)}{\text{degree}(\mathcal{V}_i)}$$

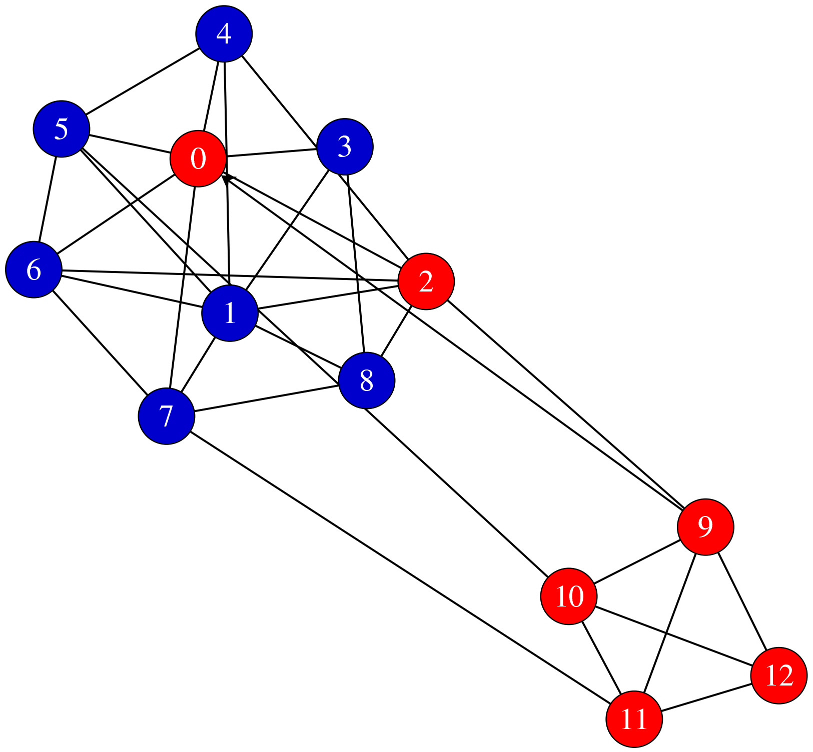
Ratio Cut: minimize $$\sum_{i=1}^{c} \frac{\text{links}(\mathcal{V}_i, \mathcal{V} \setminus \mathcal{V}_i)}{|\mathcal{V}_i|}$$

[Shi & Malik, IEEE Pattern Analysis & Machine Intelligence, 2000]
[Chan, Schlag & Zien, IEEE Integrated Circuits & Systems, 1994]

Examples



Normalized Cut



Ratio Cut

Spectral Clustering

- Take a real relaxation of the clustering objective

Spectral Clustering

- Take a real relaxation of the clustering objective
- Globally optimal solution of the relaxed problem is given by eigenvectors
  - For ratio cut: compute smallest eigenvectors of the Laplacian $L = D - A$
  - For normalized cut: compute smallest eigenvectors of the normalized Laplacian $I - D^{-1/2}AD^{-1/2}$
  - Post-process eigenvectors to obtain a discrete clustering

Spectral Clustering

- Take a real relaxation of the clustering objective
- Globally optimal solution of the relaxed problem is given by eigenvectors
  - For ratio cut: compute smallest eigenvectors of the Laplacian $L = D - A$
  - For normalized cut: compute smallest eigenvectors of the normalized Laplacian $I - D^{-1/2}AD^{-1/2}$
  - Post-process eigenvectors to obtain a discrete clustering
- Problem: Can be expensive if many eigenvectors of a very large graph are to be computed

The k-means Algorithm

- Given a set of vectors and an initial clustering, alternate between computing cluster means and assigning points to the closest mean

1. Initialize clusters π_c and cluster means $\mathbf{m}_c$ for all clusters c.
2. For every vector $\mathbf{a}_i$ and all clusters c, compute

$$d(\mathbf{a}_i, c) = \|\mathbf{a}_i - \mathbf{m}_c\|^2$$

   and

$$c^*(\mathbf{a}_i) = \text{argmin}_c \; d(\mathbf{a}_i, c)$$

3. Update clusters: $\pi_c = \{\mathbf{a} \;:\; c^*(\mathbf{a}_i) = c\}$.
4. Update means: $\mathbf{m}_c = \frac{1}{|\pi_c|} \sum_{\mathbf{a}_i \in \pi_c} \mathbf{a}_i$
5. If not converged, go to Step 2. Otherwise, output final clustering.

From k-means to Weighted Kernel k-means

- Introduce weights w_i for each point $\mathbf{a}_i$: use the weighted mean instead

From k-means to Weighted Kernel k-means

- Introduce weights w_i for each point $\mathbf{a}_i$: use the weighted mean instead
- Expanding the distance computation yields:

$$\|\mathbf{a}_i - \mathbf{m}_c\|^2 = \mathbf{a}_i \cdot \mathbf{a}_i - \frac{2\sum_{\mathbf{a}_j \in \pi_c} w_j \mathbf{a}_i \cdot \mathbf{a}_j}{\sum_{\mathbf{a}_i \in \pi_c} w_j} + \frac{\sum_{\mathbf{a}_i, \mathbf{a}_j \in \pi_c} w_j w_l \mathbf{a}_j \cdot \mathbf{a}_l}{(\sum_{\mathbf{a}_j \in \pi_c} w_j)^2}$$

From k-means to Weighted Kernel k-means

- Introduce weights w_i for each point $\mathbf{a}_i$: use the weighted mean instead
- Expanding the distance computation yields:

$$\|\mathbf{a}_i - \mathbf{m}_c\|^2 = \mathbf{a}_i \cdot \mathbf{a}_i - \frac{2\sum_{\mathbf{a}_j \in \pi_c} w_j \mathbf{a}_i \cdot \mathbf{a}_j}{\sum_{\mathbf{a}_i \in \pi_c} w_j} + \frac{\sum_{\mathbf{a}_i, \mathbf{a}_j \in \pi_c} w_j w_l \mathbf{a}_j \cdot \mathbf{a}_l}{(\sum_{\mathbf{a}_j \in \pi_c} w_j)^2}$$

- Computation can be done only using inner products of data points

From k-means to Weighted Kernel k-means

- Introduce weights w_i for each point $\mathbf{a}_i$: use the weighted mean instead
- Expanding the distance computation yields:

$$\|\mathbf{a}_i - \mathbf{m}_c\|^2 = \mathbf{a}_i \cdot \mathbf{a}_i - \frac{2\sum_{\mathbf{a}_j \in \pi_c} w_j \mathbf{a}_i \cdot \mathbf{a}_j}{\sum_{\mathbf{a}_i \in \pi_c} w_j} + \frac{\sum_{\mathbf{a}_i, \mathbf{a}_j \in \pi_c} w_j w_l \mathbf{a}_j \cdot \mathbf{a}_l}{(\sum_{\mathbf{a}_j \in \pi_c} w_j)^2}$$

- Computation can be done only using inner products of data points
- Given a *kernel* matrix K that gives inner products in feature space, can compute distances using the above formula

From k-means to Weighted Kernel k-means

- Introduce weights w_i for each point $\mathbf{a}_i$: use the weighted mean instead
- Expanding the distance computation yields:

$$\|\mathbf{a}_i - \mathbf{m}_c\|^2 = \mathbf{a}_i \cdot \mathbf{a}_i - \frac{2\sum_{\mathbf{a}_j \in \pi_c} w_j \mathbf{a}_i \cdot \mathbf{a}_j}{\sum_{\mathbf{a}_i \in \pi_c} w_j} + \frac{\sum_{\mathbf{a}_i, \mathbf{a}_j \in \pi_c} w_j w_l \mathbf{a}_j \cdot \mathbf{a}_l}{(\sum_{\mathbf{a}_j \in \pi_c} w_j)^2}$$

- Computation can be done only using inner products of data points
- Given a *kernel* matrix K that gives inner products in feature space, can compute distances using the above formula
- Objective function for weighted kernel k-means:

$$\begin{aligned} \text{Minimize } \mathcal{D}(\{\pi_{c=1}^k\}) &= \sum_{c=1}^{k} \sum_{\mathbf{a}_i \in \pi_c} w_i \|\varphi(\mathbf{a}_i) - \mathbf{m}_c\|^2 \\ \text{where } \mathbf{m}_c &= \frac{\sum_{\mathbf{a}_i \in \pi_c} w_i \varphi(\mathbf{a}_i)}{\sum_{\mathbf{a}_i \in \pi_c} w_i} \end{aligned}$$

The Weighted Kernel k-means Algorithm

- Given a kernel matrix (positive semi-definite similarity matrix), run k-means in the feature space

1. Initialize clusters π_c
2. For every vector $\mathbf{a}_i$ and all clusters c, compute

$$d(\mathbf{a}_i, c) = K_{ii} - \frac{2\sum_{\mathbf{a}_j \in \pi_c} w_j K_{ij}}{\sum_{\mathbf{a}_i \in \pi_c} w_j} + \frac{\sum_{\mathbf{a}_i, \mathbf{a}_j \in \pi_c} w_j w_l K_{jl}}{(\sum_{\mathbf{a}_j \in \pi_c} w_j)^2}$$

   and

$$c^*(\mathbf{a}_i) = \text{argmin}_c \, d(\mathbf{a}_i, c)$$

3. Update clusters: $\pi_c = \{\mathbf{a} \,:\, c^*(\mathbf{a}_i) = c\}$.
4. If not converged, go to Step 2. Otherwise, output final clustering.

Equivalence to Graph Clustering

- Surprising Theoretical Equivalence:
    - Weighted graph clustering objective is *mathematically identical* to the weighted kernel k-means objective

Equivalence to Graph Clustering

- Surprising Theoretical Equivalence:
  - Weighted graph clustering objective is *mathematically identical* to the weighted kernel k-means objective
- Follows by rewriting both objectives as trace maximization problems

Equivalence to Graph Clustering

- Surprising Theoretical Equivalence:
  - Weighted graph clustering objective is *mathematically identical* to the weighted kernel k-means objective
- Follows by rewriting both objectives as trace maximization problems
- Popular graph clustering objectives and corresponding weights and kernels for weighted kernel k-means given affinity matrix A:

Objective	Node Weight	Kernel
Ratio Association	1 for each node	$K = \sigma I + A$
Ratio Cut	1 for each node	$K = \sigma I - L$
Kernighan-Lin	1 for each node	$K = \sigma I - L$
Normalized Cut	Degree of the node	$K = \sigma D^{-1} + D^{-1} A D^{-1}$

Equivalence to Graph Clustering

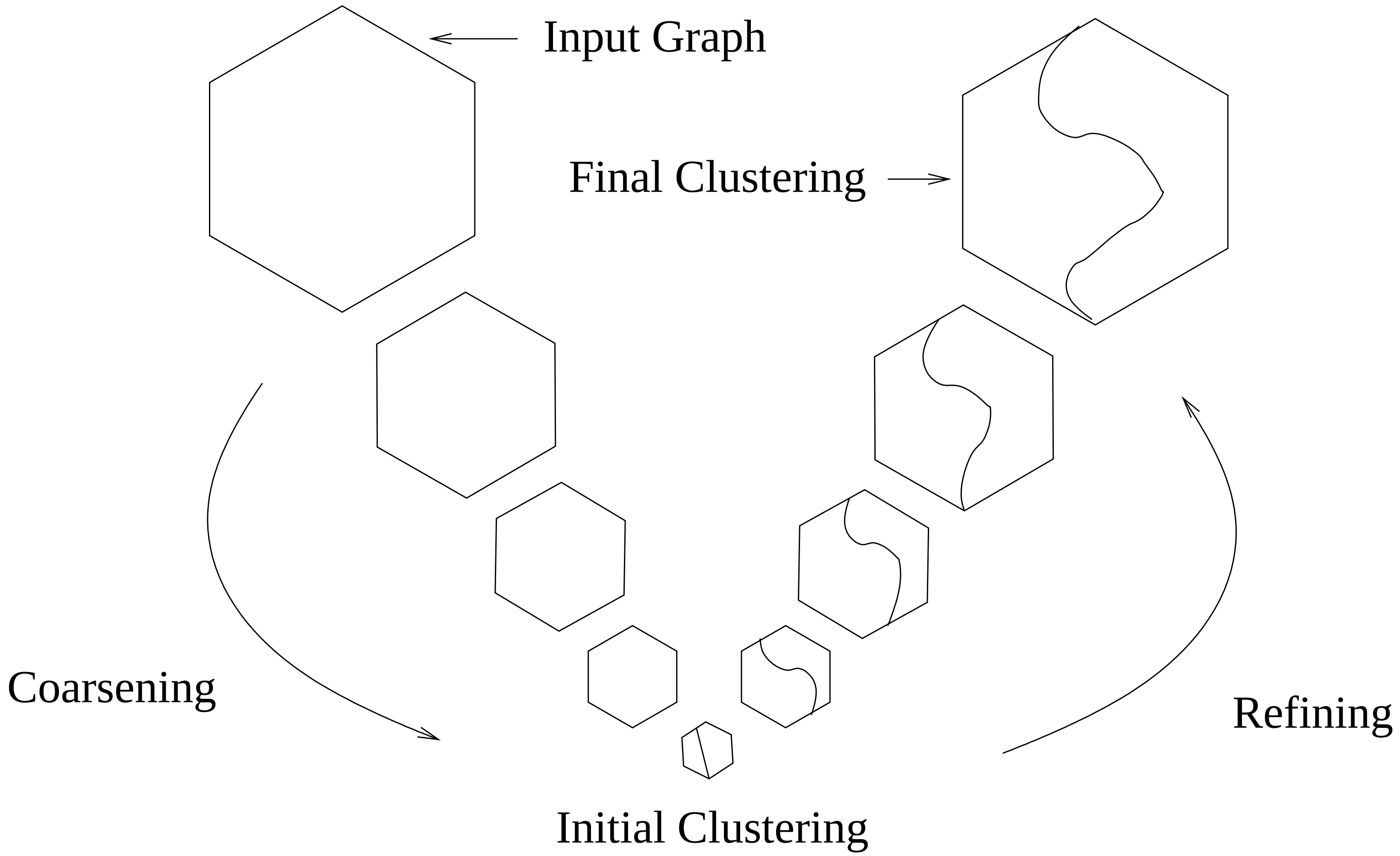
- Surprising Theoretical Equivalence:
  - Weighted graph clustering objective is *mathematically identical* to the weighted kernel k-means objective
- Follows by rewriting both objectives as trace maximization problems
- Popular graph clustering objectives and corresponding weights and kernels for weighted kernel k-means given affinity matrix A:

Objective	Node Weight	Kernel
Ratio Association	1 for each node	$K = \sigma I + A$
Ratio Cut	1 for each node	$K = \sigma I - L$
Kernighan-Lin	1 for each node	$K = \sigma I - L$
Normalized Cut	Degree of the node	$K = \sigma D^{-1} + D^{-1} A D^{-1}$

- Implication: Can minimize graph cuts such as normalized cut and ratio cut without any eigenvector computation.

The Multilevel Approach

- Overview of the approach



[CHACO, Hendrickson & Leland, 1994]
[METIS, Karypis & Kumar, 1999]

The Multilevel Approach

- Phase I: Coarsening
    - Coarsen the graph by merging nodes together to form smaller and smaller graphs
    - Use a simple greedy heuristic specialized to each graph cut objective function

The Multilevel Approach

- Phase I: Coarsening
  - Coarsen the graph by merging nodes together to form smaller and smaller graphs
  - Use a simple greedy heuristic specialized to each graph cut objective function
- Phase II: Base Clustering
  - Once the graph is small enough, perform a base clustering
  - Variety of techniques possible for this step

The Multilevel Approach

- Phase I: Coarsening
  - Coarsen the graph by merging nodes together to form smaller and smaller graphs
  - Use a simple greedy heuristic specialized to each graph cut objective function
- Phase II: Base Clustering
  - Once the graph is small enough, perform a base clustering
  - Variety of techniques possible for this step
- Phase III: Refining
  - Uncoarsen the graph, level by level
  - Use weighted kernel k-means to refine the clusterings at each level
  - Input clustering to weighted kernel k-means is the clustering from the previous level

Experiments: gene network

Mycobacterium tuberculosis gene network: 1381 genes and 9766 functional linkages.

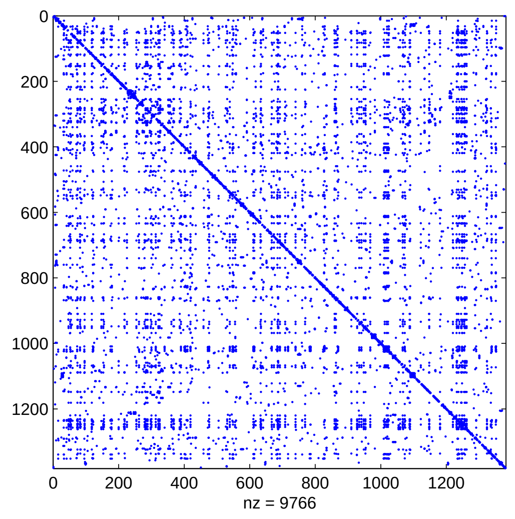
- Normalized cut values generated by Graclus and the spectral method

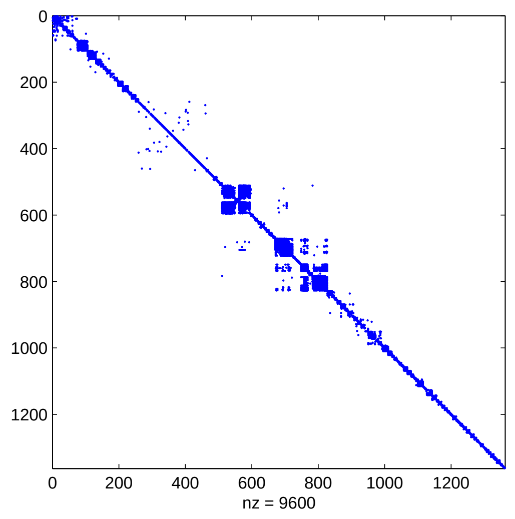
# clusters	4	8	16	32	64	128
Graclus	0	.009	.018	.53824	3.1013	18.735
Spectral	0	.036556	.1259	.92395	5.3647	25.463

Experiments: gene network

Mycobacterium tuberculosis gene network: 1381 genes and 9766 functional linkages.

- Spy plots of the functional linkage matrix before and after clustering (128 clusters)—each dot indicates a non-zero entry





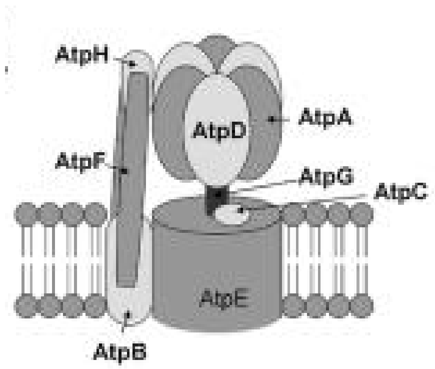
Experiments: gene network

Mycobacterium tuberculosis gene network: 1381 genes and 9766 functional linkages.

- Two example clusters: Histidine biosynthesis pathway and ATP synthase multiprotein complex

$$\begin{tikzcd}
\text{HisG} \arrow[d] \\
\text{HisE} \arrow[d] \\
\text{HisI} \arrow[d] \\
\text{HisA} \arrow[d] \\
\text{HisH} \arrow[d] \\
\text{HisF} \arrow[d] \\
\text{HisB} \arrow[d] \\
\text{HisC/HisC2} \arrow[d] \\
\text{HisB} \arrow[d] \\
\text{HisD}
\end{tikzcd}$$

(A)



(B)

Experiments: IMDB movie data set

The IMDB contains 1.4 million nodes and 4.3 million edges.

- Normalized cut values and computation time for a varied number of clusters, using Graclus and the spectral method

Normalized cut values—lower cut values are better

# clusters	2	4	8	16	32	64	128	256
Graclus	.049	.163	.456	1.39	3.72	9.42	24.13	64.04
Spectral	.00	.016	.775	2.34	5.65	-	-	-

Computation time (in seconds)

Graclus	34.57	37.3	37.96	46.61	49.93	53.95	64.83	81.42
Spectral	261.32	521.69	597.23	1678.05	5817.96	-	-	-

Experiments: IMDB movie data set

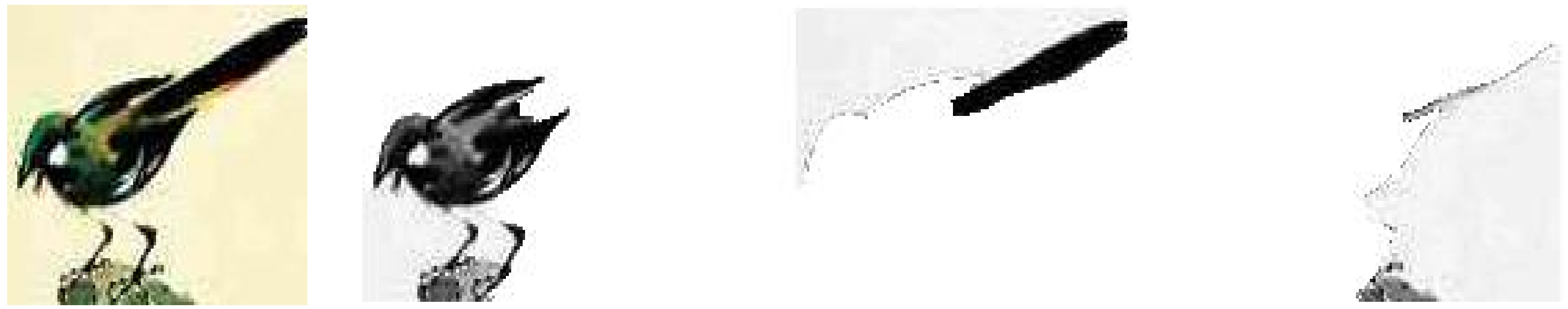
The IMDB contains 1.4 million nodes and 4.3 million edges.

- We generate 5000 clusters using Graclus, which takes 12 minutes.
- If we use the spectral method, we would have to store 5000 eigenvectors of length 1.4M; that is 24 GB main memory.

Movies	Actors
Harry Potter and the Sorcerer's Stone	Daniel Radcliffe, Rupert Grint,
Harry Potter and the Chamber of Secrets	Emma Watson, Peter Best,
Harry Potter and the Prisoner of Azkaban	Joshua Herdman, Harry Melling,
Harry Potter and the Goblet of Fire	Robert Pattinson, James Phelps,
Harry Potter and the Order of the Phoenix	Tom Felton, Devon Murray,
Harry Potter: Behind the Magic	Jamie Waylett, Shefali Chowdhury,
Harry Potter und die Kammer des Schreckens:	Stanislav Ianevski, Jamie Yeates,
Das grobe RTL Special zum Film	Bonnie Wright, Alfred Enoch, Scott Fern,
J.K. Rowling: Harry Potter and Me	Chris Rankin, Matthew Lewis, Katie Leung
	Sean Biggerstaff, Oliver Phelps

Experiments: Image segmentation

- Leftmost plot is the original image and each of the 3 plots to the right of it is a component (cluster) — body, tail and background.



- Normalized cut value for this multilevel clustering is .022138, smaller than .023944 for spectral

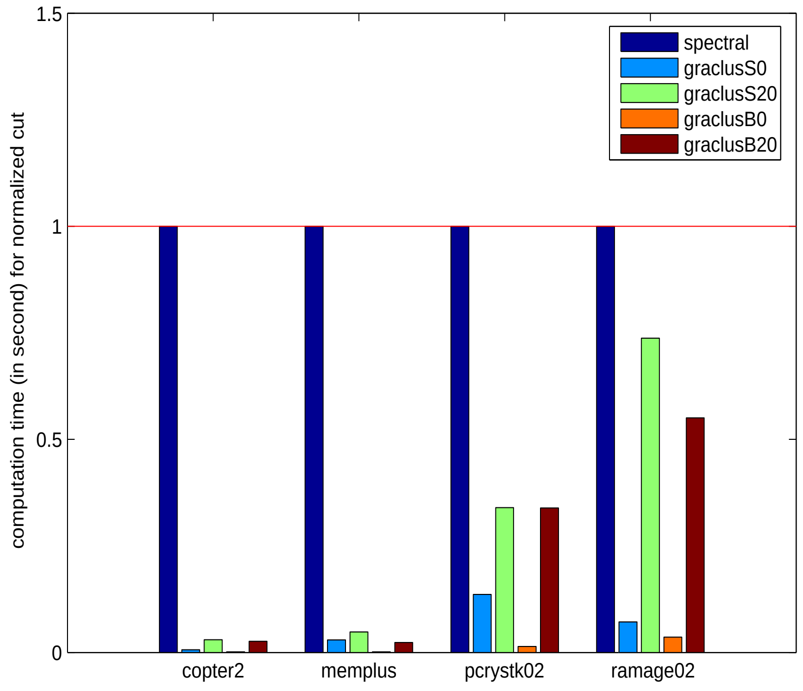
Experiments: Benchmark graph clustering

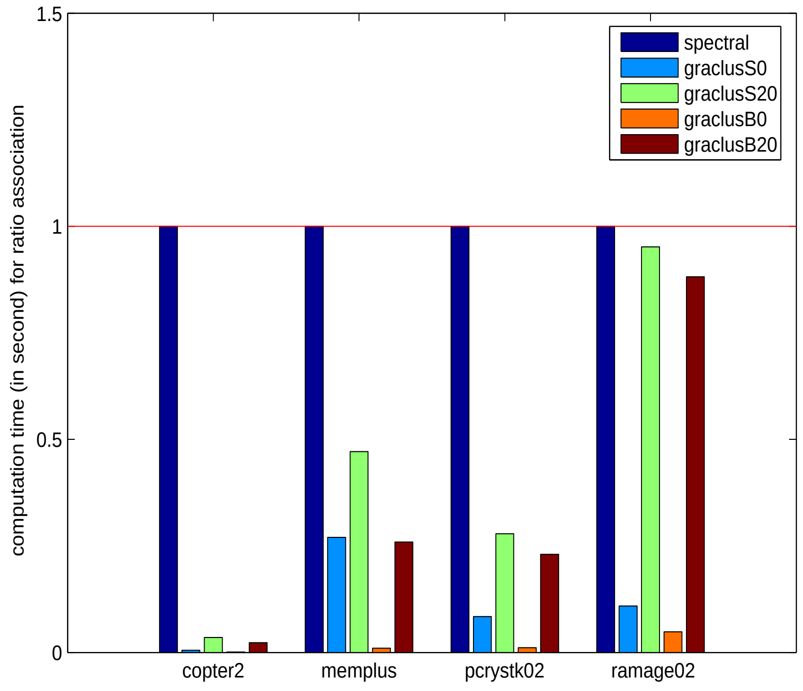
Test graphs:

Graph name	No. of nodes	No. of edges	Application
copter2	55476	352238	helicopter mesh
memplus	17758	54196	memory circuit
pcrystk02	13965	477309	structural engineering
ramage02	16830	1424761	navier stokes and continuity equations

Experiments: Benchmark graph clustering

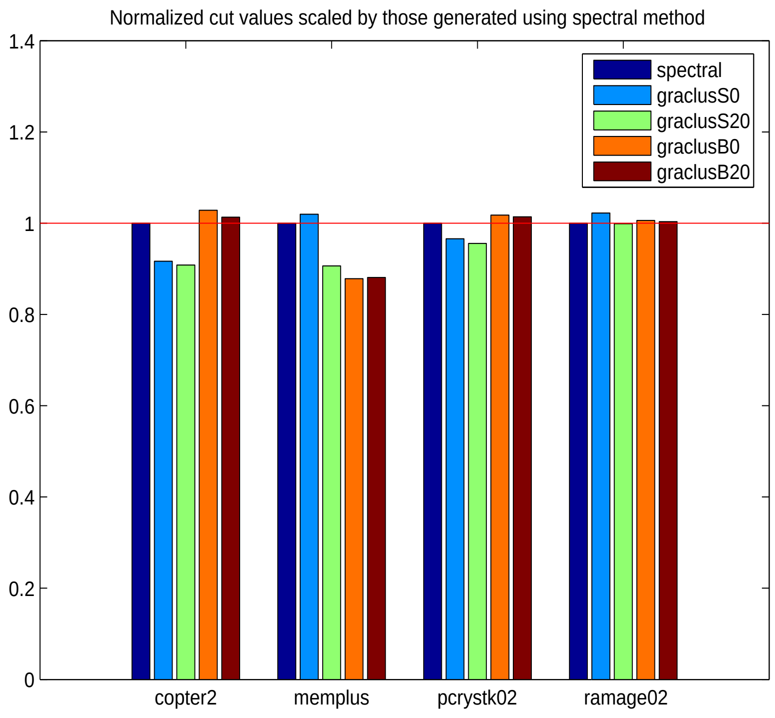
- Computation time:

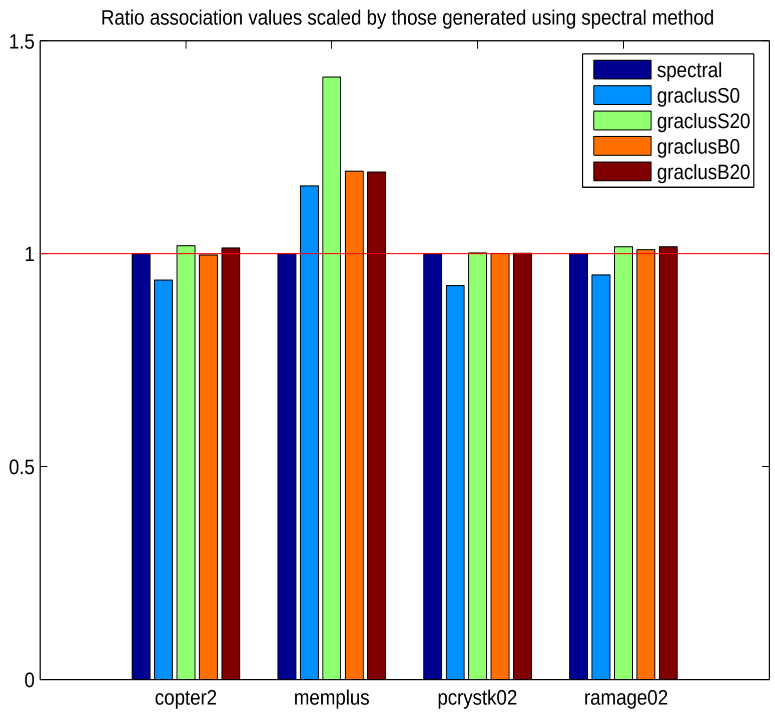




Experiments: Benchmark graph clustering

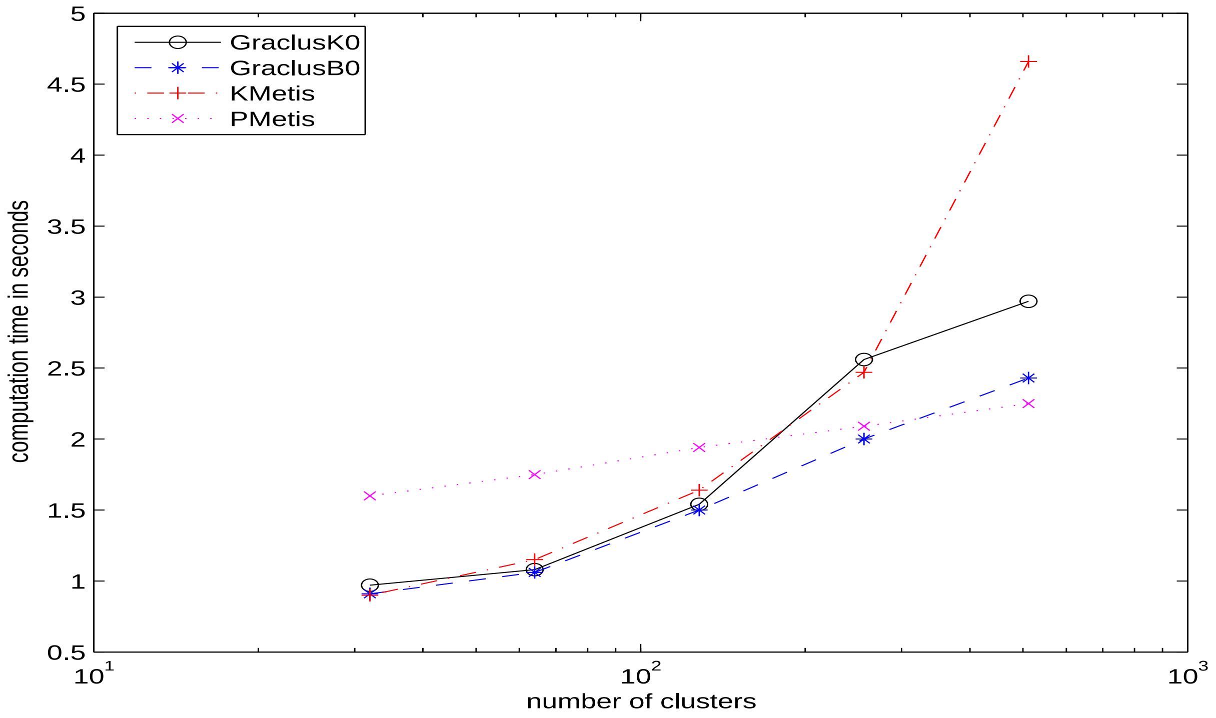
- Quality (normalized cut and ratio association):





Experiments: Benchmark graph clustering

- Computation time comparison between Graclus and Metis



Conclusions

- Minimizing graph cuts such as the normalized cut is useful in many applications
- A mathematical equivalence between spectral graph clustering objectives and the weighted kernel k-means objective
- Multilevel algorithm uses kernel k-means in its refinement phase
- Experimental results show that the multilevel algorithm, as compared to a state-of-the-art spectral clustering algorithm:
  - Mostly outperforms spectral algorithm in terms of quality
  - Significantly faster
  - Requires much less memory