DIMACS Technical Report 2008-05
April 2008

Coring Method for Clustering a Graph

by

Thang Le
Dept. of Computer Science
Rutgers University
New Brunswick, New Jersey 08903

Casimir Kulikowski
Dept. of Computer Science
Rutgers University
New Brunswick, New Jersey 08903

Ilya Muchnik
DIMACS
Rutgers University
New Brunswick, New Jersey 08903

ABSTRACT

Graph clustering partitions a graph into subgraphs with strongly interconnected nodes, while nodes belonging to different subgraphs are weakly connected. In this paper, we propose a new clustering method applicable to either weighted or unweighted graphs in which each cluster consists of a densely connected core region surrounded by a region with lower density. We have developed a highly efficient and robust method to identify nodes belonging to dense cores of clusters. The set of the nodes is then divided into groups, each of which is the representative for one cluster. These groups are finally expanded into complete clusters covering all the nodes of the graph. Experiments with both synthetic and real datasets for gene expression analysis and image segmentation yield very encouraging results.

1 Introduction

Clustering classifies unlabeled data objects by partitioning them into subsets, where every object in a subset is more similar to other objects of the same subset and less similar to the objects of other subsets. It is used frequently for exploratory data analysis for a wide range of problems in fields such as computer vision, bioinformatics, text mining, and market research. A major challenge lies in deciding which of many possible clustering criteria is likely to work best for a particular problem, given the lack of objective labels to provide validation. Usually, a criterion function measures the quality of clustering, and optimization methods are used to find a solution that maximizes or minimizes the defined criterion function. But, unfortunately, no criterion function is universally better than others, and several different clustering solutions may prove equally consistent with an expert assessment of their utility. One can always find counter examples where the optimum solution for a criterion function does not match an intuitively expected solution. Moreover, many of the criterion functions are NP-hard or very expensive to solve, so it is usual to seek approximate methods that estimate solutions to these functions.

Graph clustering methods solve the clustering problem on datasets represented by a graph, typically, a proximity graph. Let us consider an undirected graph $G = (V, E, W)$, where V is the set of nodes, E is the set of edges, W is a matrix with entry w_{ij} being the weight of the edge between nodes i and j. In proximity graphs, w_{ij} represents the degree of similarity of the objects i and j. A higher value of w_{ij} reflects a higher similarity between i and j. Proximity graphs are the natural representation for datasets from fields such as social networks, interaction networks, web hyperlink where pairwise relationships between data objects are explicitly provided. If data objects are represented in a feature space, a proximity graph can be derived by computing pairwise similarities based on Euclidean or other distance metrics between data points in the feature space, so that the dataset can be analyzed by graph-based methods.

Graph clustering methods aim to discover dense subgraphs in an unweighted graph, or highly weighted subgraphs in a weighted graph, such that the sum of edge weights inside subgraphs is high, while the sum of weights of edges connecting between different subgraphs is low. Thus, applying a graph clustering method to a proximity graph will produce a set of subgraphs, such that each subgraph corresponds to a group of similar objects, which are dissimilar to objects of groups corresponding to other subgraphs. There are different approaches for clustering a graph such as maximum cliques, thresholding minimum spanning tree, random walk, and spectral clustering. In this paper, we propose a method that finds clusters of a graph by first detecting nodes in the densest regions of clusters, then identifying the cores of clusters, and lastly assigning all other nodes to the nearest cluster. We discuss related works in Section 2. Section 3 describes in detail the new approach. Section 4 presents some results of our experiments with gene expression analysis and image segmentation. Advantages of the method and future work are discussed in Section 5.

2 Related works

In this section, we briefly summarize the layered clustering approach of B. Mirkin and I.

Muchnik [1], which has initiated the new clustering method. On a graph $G = (V, E, W)$, a linkage function $\pi(i, H)$ is defined to measure the similarity of a node i with a subset of nodes $H \subseteq V$. An example of the linkage function is the pairwise-based function defined by $\pi(i, H) = \Sigma_{j\in H} w_{ij}$. Usually this linkage function is used because of its intuitive meaning, simplicity and computation advantage. Based on the defined function π, the density of H is measured by a set function F defined as $F(H) = \min_{i\in H} \pi(i, H)$. That is, the F value of H is the minimum value of the linkage function π over all the elements of H. The largest subset H^* with the global maximum value of $F(H)$ is called the maximizer and interpreted as the subset with the highest density, i.e., $H^* = \text{argmax}_{H\subseteq V} F(H)$. It has been shown that the maximizer can be found efficiently by a greedy algorithm if the function $F(H)$ is quasi-concave, i.e., $\forall H_1, H_2$: $F(H_1\cup H_2) \geq \min(F(H_1), F(H_2))$. Furthermore, $F(H)$ is quasi-concave if the linkage function π is monotonically increasing, meaning $\forall i\in H_1\subseteq H_2$: $\pi(i, H_1) \leq \pi(i, H_2)$. Therefore, we can easily find the maximizer for $F(H)$ if we define a monotone function for π. The pairwise-based linkage $\pi(i, H) = \Sigma_{j\in H} w_{ij}$ is an example of a monotone function.

The following greedy algorithm finds the maximizer for a graph.

```
Input: Set $V$ and a monotone linkage function $\pi(i, H)$.
Output: The maximizer of $V$.
$t \leftarrow 1$.
$H_t \leftarrow V$.
While $H_t$ is nonempty.
    $F_t \leftarrow \min_{i\in Ht} \pi(i, H_t)$.
    $H_{t+1} \leftarrow H_t - \{i \mid i \in H_t \wedge \pi(i, H_t) = F_t\}$.
    $t \leftarrow t + 1$.
Return the maximizer which is the largest $H_{t^*}$ such that $F_{t^*} = \max_t F_t$.
```

The time complexity for a straightforward implementation of this procedure is $O(|V|^2\tau)$, where τ is the time for evaluating the linkage function π [8]. If we use the pairwise-based linkage function, the running time can be improved by using more efficient data structures and sorting algorithms.

Using the above procedure, the clustering method discovers clusters of a graph by iteratively finding and taking the maximizer out of the graph until it is empty [7, 8]. Each connected component in the extracted maximizers is output as a cluster of the graph.

There are several issues with the above clustering method. Firstly, if the graph is complete or highly connected, then $F(H)$ tends to decrease when a node is removed, so the maximizer tends to cover a very large part of the graph (sometimes almost the whole graph). Therefore, our expected clusters can actually exist inside the maximizer. Secondly, removing the maximizer from the graph will likely alter or even destroy the original structure of the data and make the remaining subgraph become unusable, e.g., many 'orphan' nodes can be left out when the maximizer is removed. Finally, we can not cluster a dataset at different scales. Since there may be different solutions based on different situations, we may want to cluster the same dataset from a fine-grained level to a coarse-grained level. We will illustrate above issues by a simple example. Fig. 1 shows an unweighted graph with 12 nodes and 21 edges. Intuitively, we expect a partitioning with two clusters of the same size on the left and on the right. However, the maximizer of this graph is a subgraph of 8 nodes consisting of two maximal cliques connected by an edge, so it actually contains two expected clusters and should not be regarded as one cluster. Furthermore, if the maximizer is taken out, the remaining subgraph is a set of 4 isolated nodes which then become 4 singleton clusters. Obviously, this is not an acceptable clustering solution.

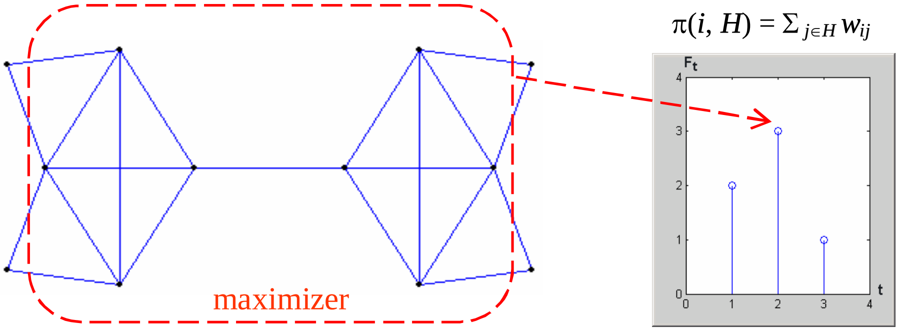


Figure 1: Limitations of clustering based on the maximizer of the graph.

3 Coring method for clustering a graph

3.1 Dense cores of clusters of a graph

We assume that every cluster of the input graph has a region of high density called a ‘cluster core’, surrounded by sparser regions (non-core). The nodes in cluster cores are denoted as ‘core nodes’, the set of core nodes as the ‘core set’, and the subgraph consisting of core nodes as the ‘core graph’. For a graph satisfying the above core assumption, the greedy procedure used in the layered clustering method has two useful properties: (*i*) the least similar node is removed at each iteration, so the nodes of the graph are arranged in an order where nodes in sparser regions are placed ahead of nodes in denser regions. In other words, the procedure progresses from outer layers to inner layers of the graph, (*ii*) density values drop significantly when the procedure removes nodes belonging to cluster cores of the graph.

In Fig. 2, we show an example of a weighted graph of 25 nodes and 126 edges with two dense cores. Edge weights are computed from the distances between the positions of two end nodes in the plane: $w_{ij} = 1 - \frac{dist(i, j)}{\max_{x,y\in V} dist(x, y)}$, where $dist(x, y)$ is the Euclidean distance between points x and y on the plane. So edge weights $w_{\mathrm{ij}} \in [0, 1]$ are inversely proportional to their lengths. In the sequence of F values, we can see that the procedure goes from nodes in the sparse regions at the boundaries to nodes in the dense regions at the center. Also, the sequence of F values has two large downhill sections corresponding to two cluster cores.

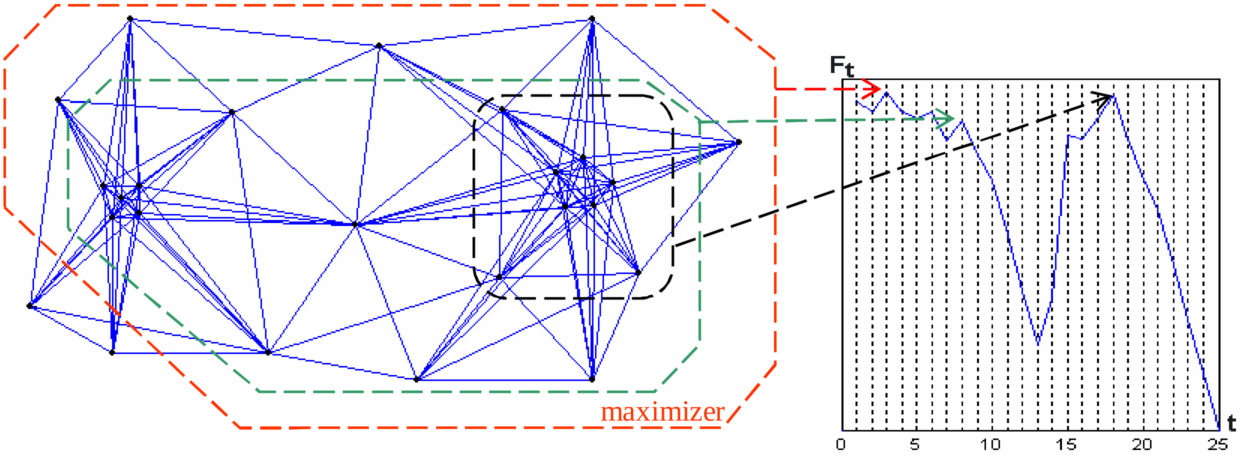


Figure 2: A graph with two clusters which have a dense core and its *F* value sequence

We have taken advantage of these properties to design a new clustering method for graphs in which each cluster has a dense core. For each node *i* of $H \subseteq V$, we define the local density at *i* with respect to *H* as:

$$d(i,H) = \frac{1}{|H|} \sum_{j \in H} w_{ij} \, .$$

Function *D*(*H*) measures the local density of the weakest node of *H* defined by:

$$D(H) = \min_{i \in H} d(i,H) \, .$$

D(*H*) can be regarded as the minimum density of *H*. The node $m = \operatorname{argmin}_{i \in H} d(i,H)$ is referred to as the weakest node of *H*. Fig. 3 shows the sequence of *D* values computed from the sequence of *F* values of the graph in Fig. 2. We can see that at the beginning, when some of the weakest nodes of the graph are removed, *D* value increases, while *F* value tends to decrease. So *D*(*H*) is better than *F*(*H*) in estimating subgraph densities.

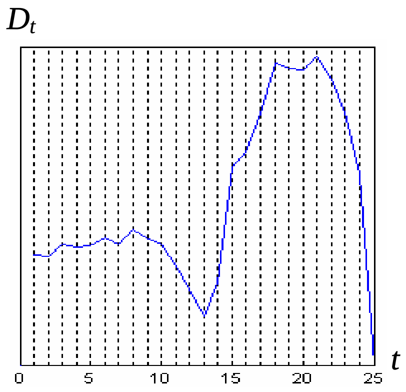


Figure 3: Minimum density variation sequence of the graph in Fig. 2.

By analyzing the variation of the minimum density value D, we can identify core nodes located in the dense cores of clusters. Specifically, if the weakest node is in a sparse region, the D value will increase when this node is removed, in other words, the next weakest node to be removed will be in a region with higher density. On the other hand, if the removal of the weakest node causes a significant drop in D value, then this node is highly connected with a set of stronger nodes in a high density region. It is potentially a core node because its removal greatly reduces the density of nodes around it. To illustrate these points, let us consider again the graph in Fig. 2. If we continuously remove the weakest node, D values tend to increase while we gradually go to the first cluster core on the left. At some point, when we remove nodes from the core (t=8→13), D values drop precipitously because the cluster core collapses. After the first cluster is fully removed, D values raise but then drop again when the second cluster core collapses (t=21→25). Thus the decreases of D values indicate core nodes of cluster cores.

3.2 Coring method for clustering a graph

We find clusters of a proximity graph in 4 steps. The coring method is outlined in the following procedure.

Input: Proximity graph G.
Output: Clustering of G.
1. Compute the sequence of density variation.
2. Identify core nodes based on the density variation and two input parameters.
3. Partition the set of core nodes into groups, each of which represents a cluster core.
4. Expand the cluster cores into full clusters.

3.2.1 Step 1: Compute the density variation sequence

This is an iterative procedure that computes the sequence of density variation. In each iteration t, we compute the minimum density D_t, and the set of the weakest nodes M_t. Then, M_t is removed from the graph. When the graph becomes empty, the procedure returns the sequences of D_ts and M_ts, which contain D values and the nodes relating to each D value, respectively.

3.2.2 Step 2: Identify the set of core nodes

Core nodes are identified based on the sequences of D_ts and M_ts and two parameters: $\alpha \in [0,1)$ and $\beta \in \mathbf{N}$. Elements of M_t are selected to be core nodes if D_t satisfies two conditions:

(1) Its rate of decrease R_t is greater than α. That is, $R_t = (D_t - D_{t+1}) / D_t > \alpha$.

(2) $\exists\, k \in \mathbf{N}$ such that $D_t \in \{D_{k+1}, D_{k+2}, ..., \mathrm{D}_{k+\beta}\}$, where $D_{k+1}, D_{k+2}, ..., D_{k+\beta}$ are β consecutive density values that also satisfy the condition (1).

Therefore, elements of M_t are core nodes if the corresponding D_t has the rate of decrease R_t greater than α, and D_t is among at least β successive D values which also have a rate of decrease greater than α. It means α controls the rate of decrease of D values of core nodes, and β controls the minimum size of a group of successive core nodes on the sequence of M_t.

Remark 1. The α parameter is an absolute value of the rate of decrease, so it is more convenient to replace it by another parameter which specifies a relative value. To do this, we sort all positive R_ts and then set α to the R_t value located at a relative position δ of the sorted list, so specifying α can be replaced by specifying δ. D_ts can also be ranked according to the sorted list of R_ts. A larger R_t indicates a greater drop in the minimum density when M_t is removed. Therefore, high ranking D_ts corresponding to large R_ts identify M_ts that contain core nodes.

3.2.3 Step 3: Partition the core set into cluster cores

The set of core nodes identified in step 2 is partitioned into groups representing cluster cores. For sparse graphs, we can find all the connected components of the core graph, and then each component is regarded as a cluster core. When the graph is highly connected, the core graph may be connected in only one component. In this case, before finding connected components, we can use a preprocessing such as: (*i*) for unweighted graphs, step 1 is applied again to the core graph to get a smaller set of core nodes, (*ii*) for weighted graphs, the histogram of edge weights can be scanned to determine a threshold in order to remove weak edges linking core groups.

In fact, step 3 is to find a solution for another clustering problem, so any graph clustering method can be used to partition the core set. For example, agglomerative hierarchical clustering is an excellent method for partitioning core nodes in a highly connected weighted graph. The dendrogram can be built for the set of core nodes. If core groups of the core set are well separated, they will be easily recognized by visualizing the dendrogram. Note that partitioning the core set is easier than clustering the original graph because we have a smaller set of nodes, and the separation of cluster cores in the core set is much better than the separation of clusters in the original graph.

3.2.4 Step 4: Expand the cluster cores into full clusters

This final step is similar to solving a supervised classification problem where unlabeled data is classified based on a training set of previously labeled data. Here we take advantage of a property of step 1 of going from outer to inner layers, in other words, nodes in the sequence M_t are ranked from sparse to dense regions. Therefore, scanning the sequence backwards will go from the centers to the boundaries of clusters. While scanning the sequence, non-core nodes are assigned to the most similar cluster, thus expanding cluster cores to full clusters. Each non-core node n of M_t is assigned to the cluster C^* = argmax$_C$ $S(n, C)$, where $S(n, C)$ measures the degree of similarity of a node n with a cluster C. For weighted graphs, it can be defined by:

(1) $S(n, C)$ = average$_{i\in C, w_{ni}>0}$ w_{ni}, or

(2) $S(n, C)$ = max$_{i\in C}$ w_{ni}.

For unweighted graphs, we can define $S(n, C)$ as:

(1) $S(n, C)$ = average$_{i\in C}$ w_{ni}, or

(2) $S(n, C) = \Sigma_{i\in C}\, w_{ni}$.

When expanding core groups, if a node m is found to be not connected to any known clusters, that is, $\forall C$: $S(m, C) = 0$, then a new cluster C^+ will be created to accommodate m. This can happen if the parameters are set to so high values that representatives of some strong clusters are excluded from the core set.

Remark 2. To reveal the structure of clusters, we can compute a confidence degree for each node when assigning it to a cluster. Let *C*1 be the most similar cluster and *C*2 be the second most similar cluster of node *n*. The confidence degree of *n* is defined as: $confidence(n) = 1 - \frac{S(n, C2)}{S(n, C1)}$.

Fig. 4 shows the clustering result for the graph in Fig. 2 with δ=40% β=2. Red points indicate core nodes. For each node, the cluster label and confidence degree are shown.

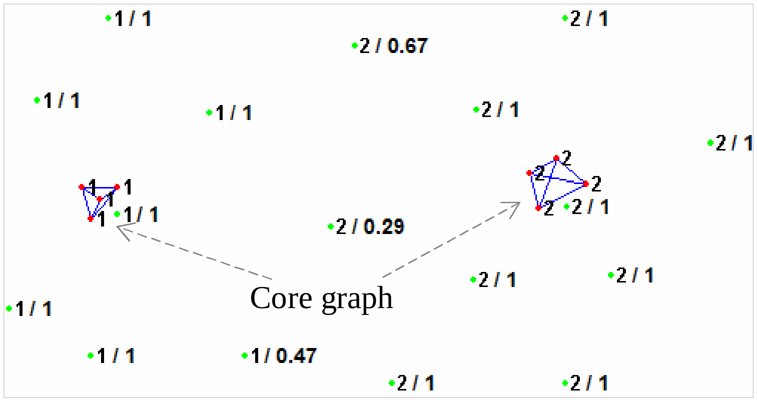


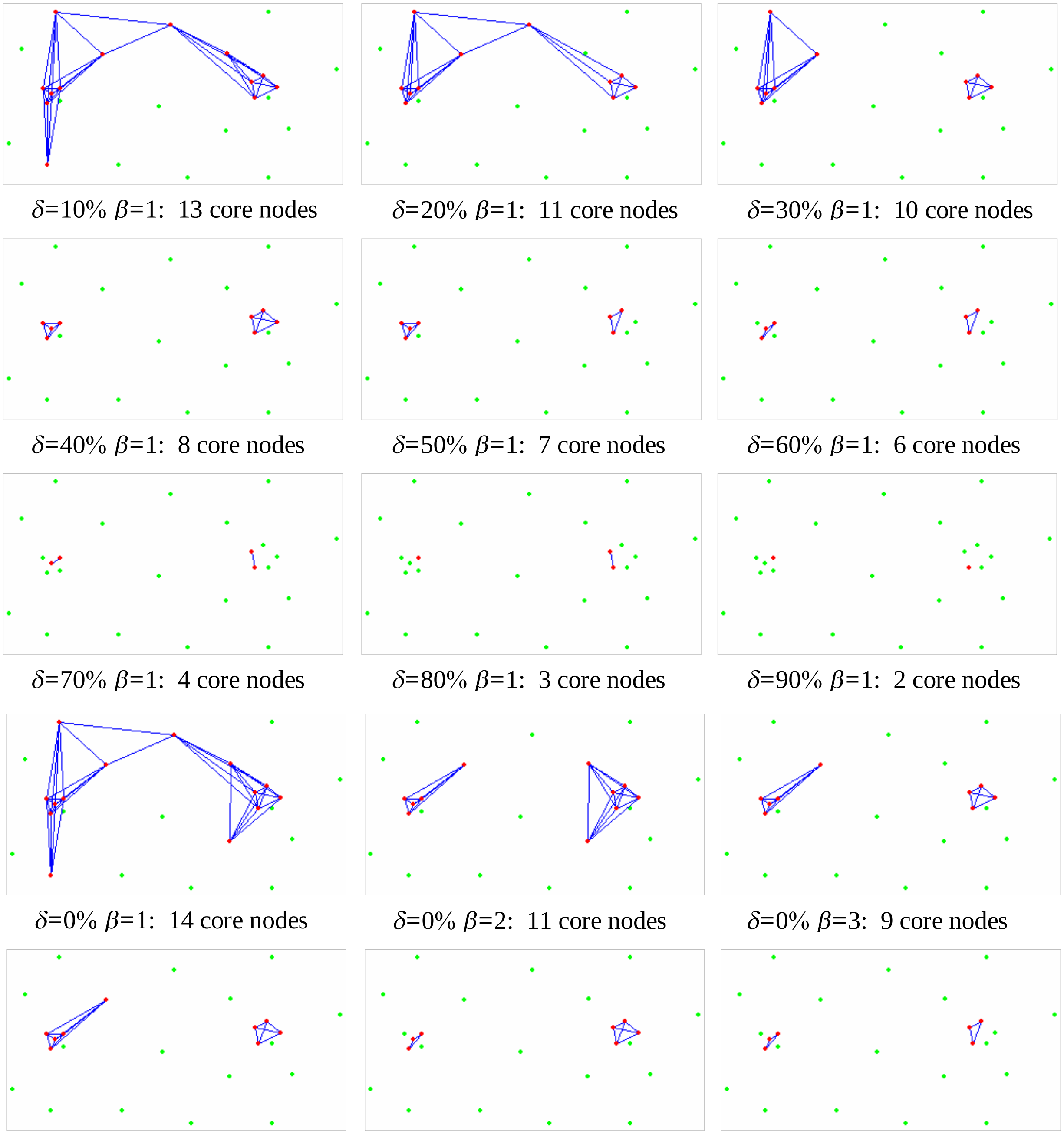
Figure 4: Cluster labels and confidence degrees of the clustering result for the graph in Fig. 2.

3.2.5 Time complexity

Using graph adjacency-list representation, the method has a similar complexity as the procedure to find the maximizer. With $\pi(i, H) = \Sigma_{j \in H} w_{ij}$ and Fibonacci heaps, step 1 can be implemented in $O(|E| + |V|\ log\ |V|)$ time [2]. Step 2 has the complexity of $O(|V|)$. In step 3, a bread-first or depth-first search to find connected components runs in $O(|V_C| + |E_C|)$, where V_C and E_C are the set of nodes and edges of the core graph, respectively. The time complexity of step 4 is $O(|E|)$. The total time complexity is therefore dominated by the complexity of step 1 which varies from $O(|V| \log |V|)$ for sparse graphs to $O(|V|^2)$ for dense graphs. An advantage of the method is step 1 has to be done only once for all the settings of δ and β. If a parameter is changed, only steps 2, 3, and 4, which are very fast, need to be re-executed to produce a new clustering result.

3.2.6 Effects of the parameters δ and β on the set of core nodes

Since the number of nodes is finite, the parameter space is discrete and there are a limited number of possible settings for parameters. In the previous example, to see how parameters affect the core set, we keep β constant and increase δ from 10% to 90%, the number of core nodes will go down from 13 to 2 nodes. With δ greater than 30%, core nodes are separated in 2 connected components. When δ is 90% there is only 1 node in each cluster core. Similarly, we keep δ constant and increase β from 1 to 3, the core set gets smaller. As we can see in Fig. 5, when δ and β are increased, the core set shrinks and at the same time, core nodes are extracted from deeper inside cluster cores. For most of the parameter settings, we get a similar clustering result with 2 clusters. Fig. 6 shows the parameter space for this example. Generally, we avoid setting the parameters to too low values because the core set may contain some nodes that are not very reliable.



δ=10% β=1: 13 core nodes

δ=20% β=1: 11 core nodes

δ=30% β=1: 10 core nodes

δ=40% β=1: 8 core nodes

δ=50% β=1: 7 core nodes

δ=60% β=1: 6 core nodes

δ=70% β=1: 4 core nodes

δ=80% β=1: 3 core nodes

δ=90% β=1: 2 core nodes

δ=0% β=1: 14 core nodes

δ=0% β=2: 11 core nodes

δ=0% β=3: 9 core nodes

δ=0% β=4: 9 core nodes

δ=40% β=3: 7 core nodes

δ=50% β=3: 6 core nodes

Figure 5: Effects of the parameters δ and β on the set of core nodes.

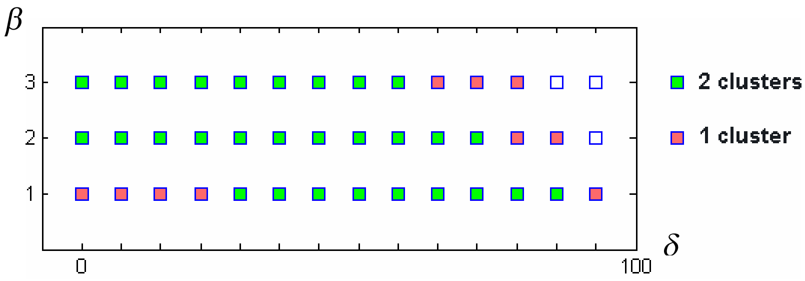


Figure 6: Parameter space for clustering the graph in Fig. 2.

4. Experiments

4.1 Gene expression analysis experiment

Clustering applications to gene expression analysis have been demonstrated in [3]. Here we tackle the problem of tissue clustering which aims to find connections between gene expressions and statuses of tissues. In other words, we want to see if it is possible to predict the status of a tissue based on its gene expressions. The dataset used in this experiment is publicly available at http://microarray.princeton.edu/oncology/affydata/index.html. It contains 62 samples including 40 tumor and 22 normal colon tissues. Each sample consists of a vector of 2000 gene expressions. We will set aside the sample labels (tumor/normal) and cluster the samples based on the similarities between their gene expressions. Ideally, we want to partition the sample set into two clusters such that one contains only tumor tissues and the other contains only normal tissues.

The proximity graph constructed from the gene expression vectors is a complete graph of 62 nodes. Because relative values are more important than absolute values in gene expressions, edge weights that reflect the pairwise similarities of samples are computed based on the Pearson correlation coefficient [5]. Specifically, the weight function is defined by:

$$w_{ij} = \frac{1}{2000}\sum_{k=1}^{2000}\frac{\left(i_k - m_i\right)\left(j_k - m_j\right)}{s_i s_j},$$

where i_k and j_k are gene expressions of samples i and j; m_i, m_j, s_i, s_j are means and standard deviations of i_ks and j_ks.

With δ=40% and β=2, steps 1 and 2 of the coring method identify 12 core nodes. The dendrogram of these core nodes shown in Fig. 7 exposes two well-separated groups, one contains 10 nodes and the other has 2 nodes. Thus, we cut the dendrogram at the height of 0.1 to obtain two cluster cores. Expanding these cluster cores yields two clusters. One has 40 samples consisting of 37 tumor and 3 normal tissues. The other contains 22 samples consisting of 3 tumor and 19 normal tissues.

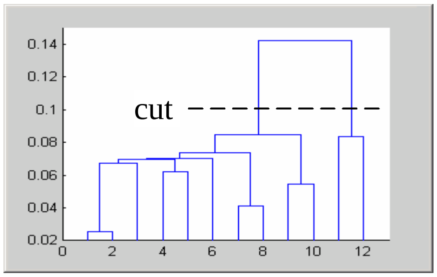
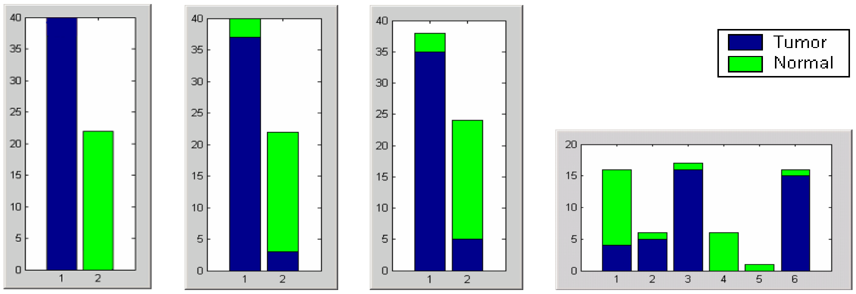


Figure 7: Dendrogram of the core graph in the gene expression analysis experiment.

Fig. 8 shows the comparison of clustering results by the coring method, [3] and [4]. The result of [4] consists of 6 clusters, but joining clusters 1, 4, 5 into one group of normal tissues and 2, 3, 6 into another group of tumor tissues will yield a clustering similar to the result in [3].



Ideal Coring [3] [4]

Figure 8: Comparison of clustering results in the gene expression analysis experiment.

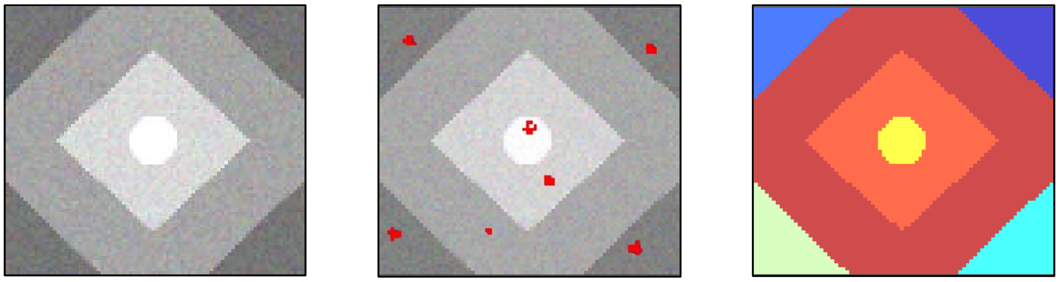
4.2 Image segmentation experiment

In this experiment, we consider the problem of partitioning a grayscale image into regions of nearby pixels that have similar intensities. Based on an image, we can construct a proximity graph where each node represents a pixel. The weight of an edge between two nodes reflects the pairwise similarity of the corresponding pixels, i.e., the likelihood that those two pixels belong to the same segment. Therefore, edge weights are computed based on the intensities and locations of pixels. Specifically, we use the following weight function described in [6]:

$$w_{ij} = \begin{cases} e^{-\left(\frac{I(i)-I(j)}{\sigma_I}\right)^2 - \left(\frac{dist(i,j)}{\sigma_d}\right)^2} & \text{if } dist(i,j) < r \\ 0 & \text{otherwise} \end{cases} ,$$

where $I(i) \in [0, 1]$ is the intensity of pixel i; $dist(i, j)$ is the Euclidean distance in pixels between i and j. There is an edge linking two nodes only if the distance between the corresponding pixels is less than r pixels, so the proximity graph is very sparse as $\pi r^2 << |V|$. We typically set $\sigma_I = 0.07$, $\sigma_d = 8$, and $r = 11$ for natural images of sizes less than 200×300. By using this weight function, strong edges exist between nodes whose corresponding pixels have the same intensity and are close to each other. So pixels inside a region with homogeneous intensity have their nodes in the proximity graph strongly connected. On the other hand, pixels at boundaries have neighbor pixels of different intensity, so their corresponding nodes are weakly connected.

The segmentation of an image is obtained by clustering its proximity graph. Nodes of each cluster of the graph correspond to pixels of a segment in the image. In Fig. 9, (a) shows a grayscale image consisting of 7 regions of nearby pixels with similar intensities. We build and cluster the image proximity graph with the coring method. Red pixels in (b) show the location of the pixels corresponding to core nodes with δ=98% and β=3. The set of core pixels consists of 7 connected components representing the cores of 7 segments. Expanding these cores gives us the expected segmentation in (c) where different segments are labeled with different colors.



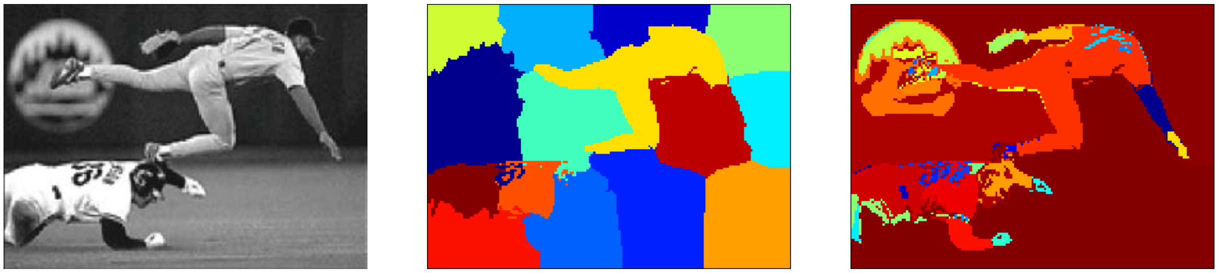
(a) (b) (c)

Figure 9: Coring method segments an image by clustering its proximity graph.

A well known graph clustering method that has been used to solve the image segmentation problem is the Normalized cut method [6]. Basically, this method partitions a graph into subgraphs based on the minimum Normalized cut, which in case of two-way cut is defined by:

$$Ncut(A,B) = \frac{\sum_{u\in A, v\in V\setminus A} w_{uv}}{\sum_{u\in A, v\in V} w_{uv}} + \frac{\sum_{u\in B, v\in V\setminus B} w_{uv}}{\sum_{u\in B, v\in V} w_{uv}},$$

where A and B are subgraphs of G. Finding the minimum Normalized cut of a graph is NP-hard, but an approximation solution for the two-way cut can be estimated using the eigenvector of the second smallest eigenvalue of the normalized Laplacian matrix $L = I - D^{-1}W$, where D is the diagonal matrix of vertex degrees. An implementation of the method developed by its author is available on the web at http://www.cis.upenn.edu/~jshi/software/. In fact, a serious problem with this method is that the normalizing factors in the criterion function make it favor balanced clusters. In Fig. 10, (a) shows a grayscale baseball image, (b) shows a segmentation result by the Normalized cut method where we can see large segments tend to break into several parts of similar sizes. Clustering on the same proximity graph, the segmentation by the coring method shown in (c) is much better. Our method also has a better performance by just a straightforward implementation. For instance, the coring method took 5.5 seconds vs. about 20 seconds for the Normalized cut method for clustering a weighted graph of $50*10^3$ nodes and $8*10^6$ edges.



(a) (b) (c)

Figure 10: Image segmentation comparison.

5 Conclusions

The coring method proposed in this paper has several advantages. It is a simple and fast clustering method working on both unweighted and weighted graphs. In terms of flexibility, we are able to manipulate core nodes and control the number of core nodes using two parameters. Core nodes produced by step 1 come from the centers of clusters so they represent informative data objects. Furthermore, because of using core nodes the method is fairly robust to noise. When adding noise to an image, we have seen that the set core nodes and segmentation results remain stable. Another advantage is that cluster structures might be revealed through the ranking of core nodes and the confidence degree of other nodes. In comparison with the approach based on the maximizer, the method has a better time complexity because the time-consuming greedy procedure is run only once for all parameter settings. It can cluster highly connected graphs where the maximizer usually covers almost the whole graph. As it does not remove any part of the graph, it does not change the graph structure and therefore does not create orphan nodes. In addition, it provides control parameters for clustering adjustment and allows human intervention and verification in the course of clustering (steps 2 and 3).

Our method works best if clusters of the graph have a strong dense core. If some cluster does not have a core or only has a faint core, it may be omitted due to not having representatives in the core set. The range of parameter settings for good clustering results is different for different graph structures. The best range is currently learnt empirically. However, sometimes we have prior knowledge about the data. For example, we may know beforehand the number of clusters. This information can be used to search the parameter space to infer good settings for parameters. It can also help the partitioning of the core set for complete or highly connected graphs.

In the future, we will study the possibility of relaxing the dense core assumption or detecting its violations. Currently the pairwise based linkage function is used, we can test the method with new linkage functions such as the consistent triplets [7] to exploit higher level relation of data objects or the linkage function for multipartite graph clustering [8]. We will look for new methods of expanding cluster cores to get better boundaries between clusters as well as ways to detect cluster structures and recursively apply the method to cluster multi-scale data. Finally, the method can be extended to cluster directed graphs. We just need to create a linkage function that takes into account directions of edges when measuring similarities in directed graphs.

References

[1] B. Mirkin and I. Muchnik, Layered clusters of tightness set functions, *Applied Mathematics Letters*, 15:147-151, 2002.

[2] M. Charikar, Greedy approximation algorithms for finding dense components in a graph, *Lecture Notes in Computer Science*, Springer-Verlag, 1913:84-95, 2000.

[3] U. Alon, N. Barkai, D. Notterman, K. Gish, S.Ybarra, D. Mack and A. Levine, Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide array, *Proc. Natl. Acad. Sci. USA*, 96(12):6745-6750, 1999.

[4] A. Ben-Dor, R. Shamir and Z. Yakhini, Clustering gene expression patterns, *Journal of Computational Biology*, 1999.

[5] D. Jiang, C. Tang and A. Zhang, Cluster Analysis for Gene Expression Data: A Survey, *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370-1386, 2004.

[6] J. Shi and J. Malik, Normalized cuts and image segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888-905, 2000.

[7] H. Yun, *Consistent triplets in graph clustering for protein sequence analysis*, PhD thesis, Department of Computer Science, Rutgers University, 2006.

[8] A. Vashist, C. Kulikowski and I. Muchnik, Ortholog clustering on a multipartite graph, *IEEE Transactions on Computational Biology and Bioinformatics*, 4(1):17-26, 2007.

Coring method for clustering a graph

Thang Le
Department of Computer Science
Rutgers University

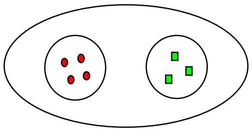
2007

Outline

- Introduction
- Layered clustering by quasi-concave set function optimization
- The coring method for clustering an undirected graph
- Preliminary experimental results
- Method advantages and issues
- Future work
- Conclusions

Introduction

- **Clustering**: Partition a set into subsets such that every element of a subset is more similar to other elements of the same subset and less similar to elements of other subsets



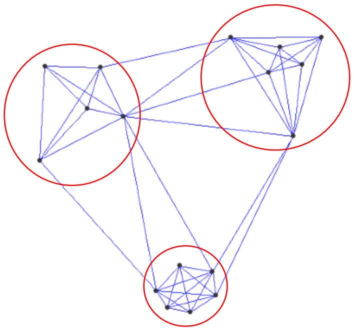
- Many clustering applications in data mining, bioinformatics, computer vision, market research, VLSI design, etc.
- Graph-based clustering methods work on graphs – a popular and powerful form of data representation

Proximity graphs

- Undirected graph $G = (V, E, W)$.
  $i, j \in V$: $w_{ij} \in W$ is the weight of edge $(i\ j) \in E$, $w_{ij} = 0$ if $(i\ j) \notin E$,
  $w_{ij} \in \{0,1\}$ for unweighted graphs, $w_{ij} \geq 0$ for weighted graphs
- **Proximity graphs**: Edge weight w_{ij} represents the degree of **similarity** between data objects corresponding to nodes *i* and *j*
- Proximity graph is a natural representation for data in fields such as social networks, interaction networks, or web hyperlink data
- If data is represented in a feature space, a proximity graph can be derived from pairwise distances between data points in the feature space, so graph-based methods can be applied for data analysis

Graph clustering

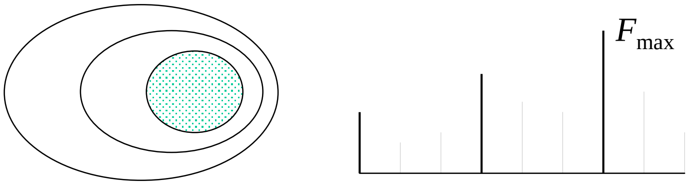
- Discover strongly or densely connected subgraphs that are weakly or sparsely connected to each other
- Clustering a proximity graph results in subgraphs corresponding to subsets of similar objects
- Many practical approaches: Layered clustering, spectral clustering, random walks, thresholding MST, maximum cliques, etc.



Layered clustering

[B. Mirkin, I. Muchnik]

- The density of a set is measured by a set function *F* which is the minimum value of linkages between the set and its elements
- The subset with the global maximum value of *F* is referred to as the densest subset of the set (the maximizer)
- Layered clusters are chain-nested subsets such that the *F* value of each subset is greater than the *F* values of subsets which are not part of it

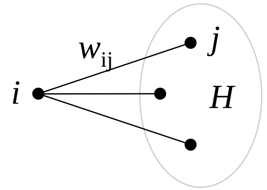


Clustering by quasi-concave function optimization

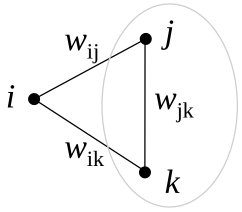
- Linkage function $\pi(i, H)$ measures the similarity of a node i with a subgraph H. Generally, $\pi(i, H) = \Sigma_{j \in H} w_{ij}$
- Set function $F(H)$ measures the proximity of the nodes of H:
  $$F(H) = \min_{i \in H} \pi(i, H)$$
- Subgraph H^* is a **maximizer** for a graph if $H^* = \operatorname{argmax}_{H \subseteq V} F(H)$
- The largest maximizer is discovered and extracted as one cluster. Repeat the extraction on the remaining graph until it is empty
- The largest H^* can be found efficiently if $F(H)$ is 'quasi-concave':
  i.e., $\forall\, H_1, H_2\colon\ F(H_1 \cup H_2) \geq \min(F(H_1), F(H_2))$
- $F(H)$ is quasi-concave iif $\pi(i, H)$ is monotonically increasing:
  i.e., $\forall\, i \in H_1 \subseteq H_2\colon\ \pi(i, H_1) \leq \pi(i, H_2)$

Examples of monotone linkage functions

- Pairwise-based linkage functions:
  - $\boldsymbol{\pi(i, H) = \Sigma_{j \in H} w_{ij}}$
  - $\pi(i, H) = \max_{j \in H} w_{ij}$

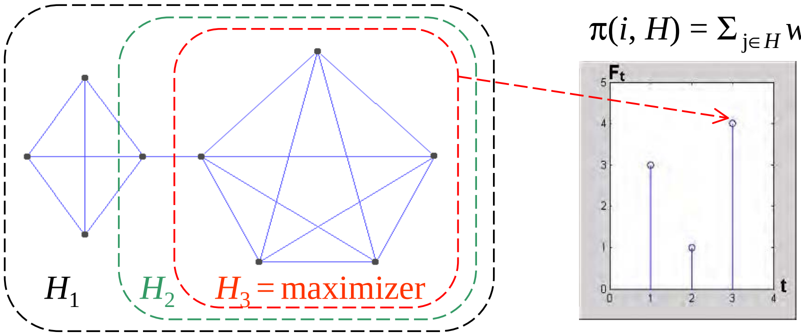


- Consistent triplet linkage function [J. Yun et al.]:
  - To exploit higher-level relation among data objects
  - $i, j, k \in V$: $\{i, j, k\}$ is a consistent triplet if $w_{ij}, w_{ik}, w_{jk} \geq \theta > 0$ and i, j, k satisfy an additional constraint such as they share a common property, i.e., higher-level relation
  - $\pi(i, H)$ = number of consistent triplets involving i in H



Procedure to find the largest maximizer

```
$t \leftarrow 1$
$H_t \leftarrow V$
**While** $H_t$ is nonempty
        $F_t \leftarrow \min_{i \in H_t} \pi(i, H_t)$
        $H_{t+1} \leftarrow H_t - \{i \mid i \in H_t \wedge \pi(i, H_t) = F_t\}$
        $t \leftarrow t + 1$
**Return** the maximizer is the largest $H_{t^*}$ such that $F_{t^*} = \max_t F_t$
```



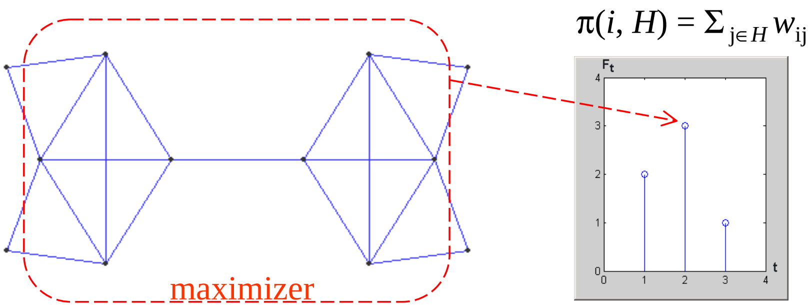
$\pi(i, H) = \Sigma_{j \in H} w_{ij}$

Time complexity

- A straightforward implementation of the procedure runs in $O(|V|^2 \tau)$ where τ is time for evaluating linkage function π [A. Vashist et al.]
- The overall complexity to extract all clusters in a graph is $O(n|V|^2 \tau)$ where n is the number of times applying the procedure
- With $\pi(i, H) = \Sigma_{j \in H} w_{ij}$ the procedure running time can be reduced:
  - $O(|E| + |V|) = O(|E|)$ for unweighted graphs using count sort
  - $O(|E| + |V| \log|V|)$ for weighted graphs using Fibonacci heaps

Limitations of clustering based on the maximizer

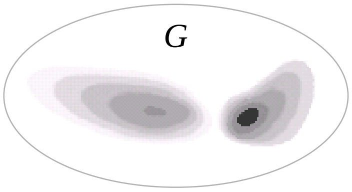
- For highly connected graphs, the maximizer covers all the graph
- Expected clusters (maximal cliques) are inside the maximizer
- ‘Orphan’ nodes are left out when the maximizer is removed
- Lack of a control parameter to cluster a dataset at different scales



$\pi(i, H) = \Sigma_{j \in H} w_{ij}$

Dense core assumption

- **Assumption:** ***Given a graph, every cluster in it has one dense core surrounded by sparser regions***



- **Cluster core**: The dense core of a cluster
- **Core nodes**: Nodes in cluster cores
- **Core set**: The set of core nodes
- **Core graph**: The subgraph consisting of core nodes

Example: Weighted graph with cluster cores

A weighted graph of 25 nodes and 126 edges:

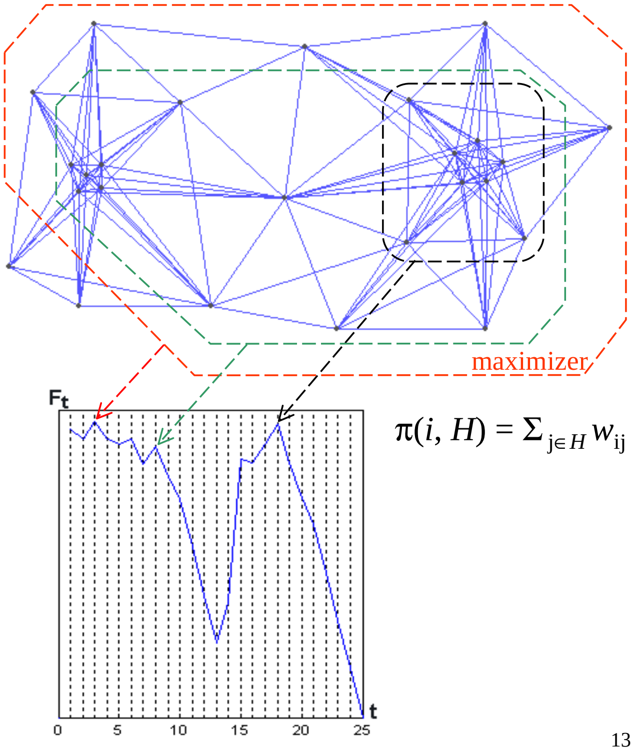
$$w_{ij} = \frac{[\max_{x,y\in V} d(x,y)] - d(i,j)}{\max_{x,y\in V} d(x,y)}$$

where $d(x, y)$ is Euclidean distance between x and y

Therefore, $w_{ij} \in [0, 1]$

Important properties:

- F_t sequence progresses from outer to inner layers
- Large downhill sections correspond to cluster cores

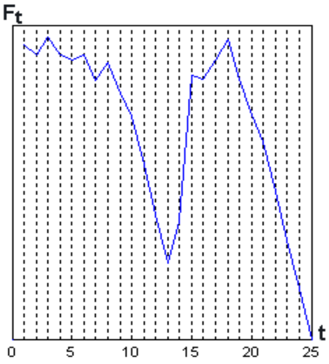


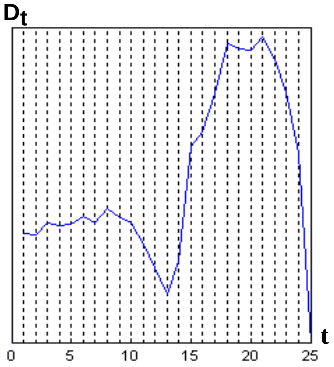
$\pi(i, H) = \Sigma_{j\in H} w_{ij}$

Minimum density measurement

- $\pi(i, H) = \Sigma_{j\in H} w_{ij}$ can measure the density at node i in subgraph H
- Define $D(H)$ to measure the minimum density of subgraph H
  $D(H)$ is $F(H)$ normalized by the size of H:

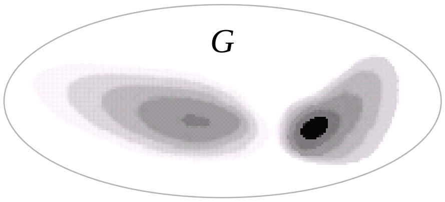
$$D(H) = \frac{F(H)}{|H|} = \frac{\min_{i\in H} \pi(i,H)}{|H|} = \frac{\min_{i\in H} \sum_{j\in H} w_{ij}}{|H|}$$

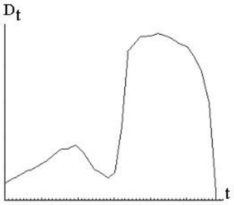




How to identify core nodes

- Core nodes can be identified by analyzing the change of minimum density value *D* while continuously removing the weakest node
- If the weakest node is in a sparse region, then *D* value will increase when the node is removed, i.e., the next node has a higher density
- If the removal of the weakest node causes a significant drop in *D* value, then the node is highly connected with a set of stronger nodes in a dense region. It is potentially a core node since its removal greatly reduces the density of nodes connecting to it





Coring procedure to cluster a proximity graph

Input: Proximity graph *G* and input parameters

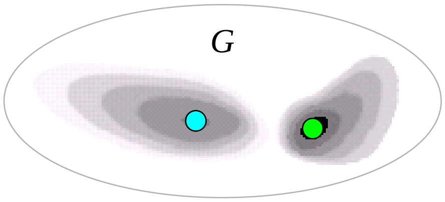
Output: Clustering of *G*

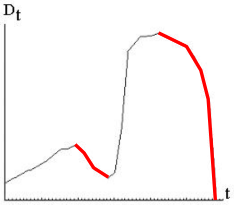
Step 1: Compute the sequence of density variation

Step 2: Identify core nodes according to the input parameters

Step 3: Partition the set of core nodes into groups (cluster cores)

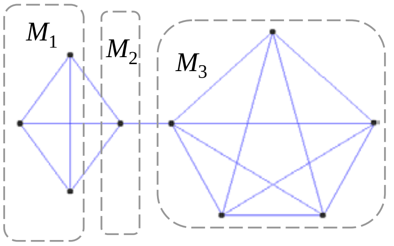
Step 4: Expand the cluster cores into full clusters





Step 1: Compute the density variation sequence

$\pi(i, H) = \Sigma_{j \in H} w_{ij}$



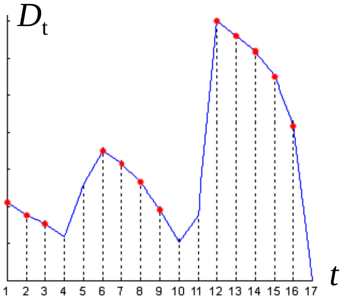
```
$t \leftarrow 1$
$H \leftarrow V$
While  $H$ is nonempty
        $F \leftarrow \min_{i \in H} \pi(i, H)$
        $D_t \leftarrow F \div |H|$
        $M_t \leftarrow \{i \mid i \in H \wedge \pi(i, H) = F\}$
        If  $M_t$ consists of more than one connected component
            then  $M_t \leftarrow$ the smallest connected component
        If  $|M_t| > 1$  and  there's no edge connecting $M_t$ with $H - M_t$
            then  remove one node from $M_t$
        $H \leftarrow H - M_t$
        $t \leftarrow t + 1$
Return  sequences of $D_t$ and $M_t$ with $t = < 1 , 2 , ... , T{-}1 , T >$
```

Step 2: Identify core nodes

- Two control parameters: $\alpha \in [0, 1)$ and $\beta \in \mathbb{N}$
- Elements of M_t are core nodes if D_t satisfies two conditions:

  (1) Rate of decrease: $R_t = (D_t - D_{t+1}) \div D_t > \alpha$

  (2) $\exists k$: $D_t \in \{D_{k+1}, D_{k+2}, ... , D_{k+\beta}\}$, where $D_{k+1}, D_{k+2}, ... , D_{k+\beta}$ are β **consecutive** density values that also satisfy condition (1)

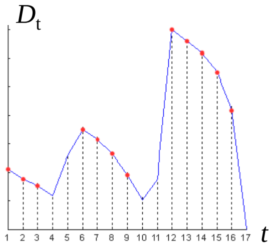


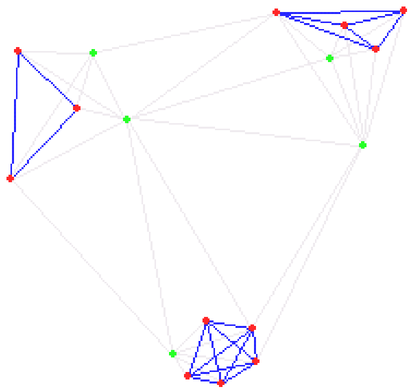
Red D_ts satisfy conditions (1) and (2) with $\alpha = 0$ and $\beta = 1 \mid 2 \mid 3$

- Sort the list of R_ts (>0), so D_ts are **ranked**. Set α to the R_t value located at a relative position δ on the sorted list. So α can be replaced by δ

Step 3: Partition core set into cluster cores

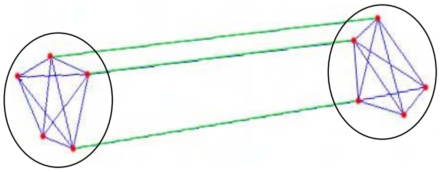
- Partition the core set into groups, each group represents a cluster core
- For sparse graphs, find connected components of the core graph, each component is considered as a cluster core

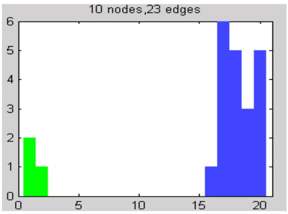




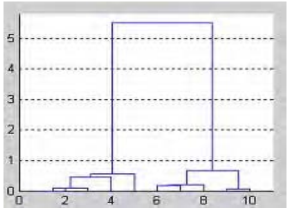
Partition core set for highly connected graphs

- Core graph may be connected in one component
  - Unweighted graphs: Apply again steps 1 and 2 to the core graph so as to get a smaller core set
  - Weighted graphs: Visualize the histogram of edge weights and find a threshold to remove weak edges between cluster cores





- Agglomerative hierarchical clustering is another good method to partition core nodes:
  - A dendrogram is built and visualized
  - Cut the dendrogram to get cluster cores



Step 4: Expand cluster cores

- Since M_t sequence built in step 1 goes from outer to inner layers:

```
For  $t = < T , T–1 , ... , 2 , 1 >$
    If  $M_t$ contains non-core nodes
        then  assign nodes of $M_t$ to the most similar cluster
```

- Assign node n to cluster $C^* = \text{argmax}_C\, S(n, C)$, where $S(n, C)$ measures the degree of similarity of node n with cluster C
- $S(n, C)$ can be defined by one of these functions:
  - $\Sigma_{i \in C}\, w_{ni}$
  - $\max_{i \in C}\, w_{ni}$ (for weighted graphs only)
  - $\text{average}_{i \in C}\, w_{ni}$ i.e. $(\Sigma_{i \in C}\, w_{ni}) \div |C|$
  - $\text{average}_{i \in C \wedge w_{ni} > 0}\, w_{ni}$ (for weighted graphs only)

Cluster structure uncovering

- To reveal cluster structure, a confidence degree can be computed for non-core nodes using their similarities with the two most similar clusters
- Let $C1$ be the most similar cluster and $C2$ be the second most similar cluster of node n. The confidence degree of n is defined by:

$$confidence(n) = \frac{S(n, C1) - S(n, C2)}{S(n, C1)}$$

- Further, $S(n, C)$ may be computed taking into account confidence degrees of nodes of C, e.g., $S(n, C) = \max_{i \in C}\, [w_{ni} \cdot confidence(i)]$

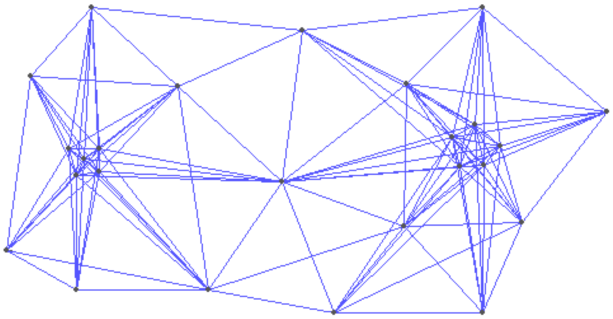
Time complexity of the method

- Time complexity of implementations on graphs with adjacency-list representation:
  - Step 1: $O(|E| + |V|\ log|V|)$ for weighted graphs (similar to the complexity of finding the maximizer)
  - Step 2: $O(|V|)$
  - Step 3: $O(|E_c|)$ where E_c is the set of edges of the core graph
  - Step 4: $O(|E|)$
- The total time is dominated by $O(|E| + |V|\ log|V|)$ of step 1 which is executed only once for all settings of parameters δ and β

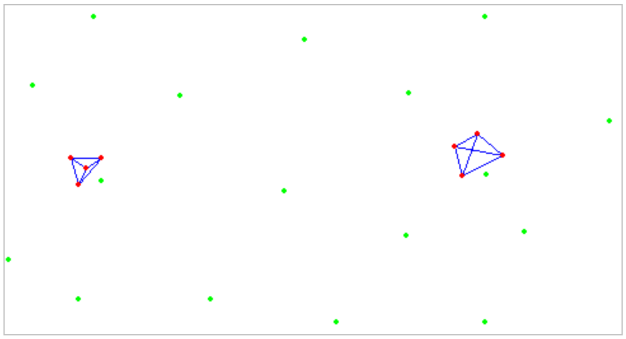
Example: Cluster a weighted graph

Cluster a weighted graph of 25 nodes and 126 edges:

$$w_{ij} = \frac{[\max_{x,y\in V} d(x,y)] - d(i,j)}{\max_{x,y\in V} d(x,y)}$$



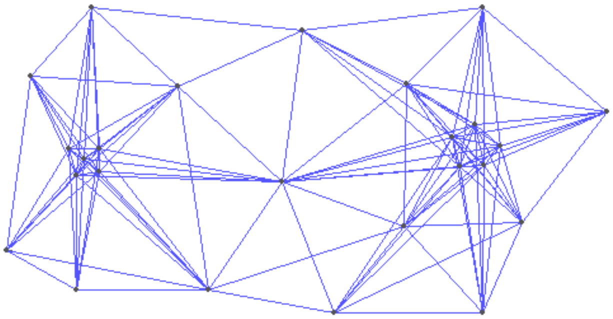
δ = 40%
β = 1
8 core nodes separated in
2 connected components



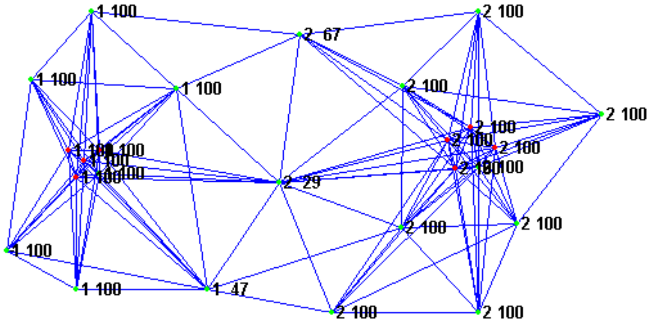
Example: Cluster a weighted graph

Cluster a weighted graph of 25 nodes and 126 edges:

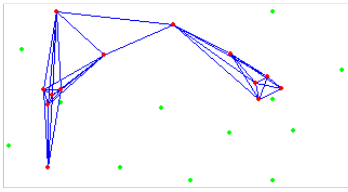
$$w_{ij} = \frac{[\max_{x,y\in V} d(x,y)] - d(i,j)}{\max_{x,y\in V} d(x,y)}$$



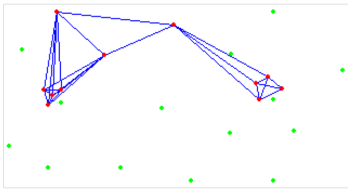
$\delta = 40\%$
$\beta = 1$
8 core nodes separated in
2 connected components



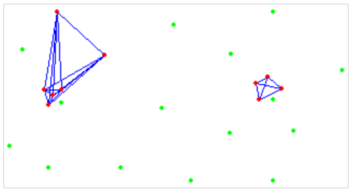
Effects of the parameters



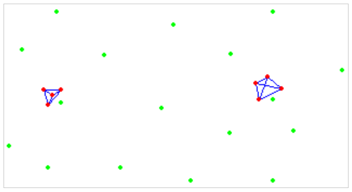
δ=10% β=1: 13 core nodes



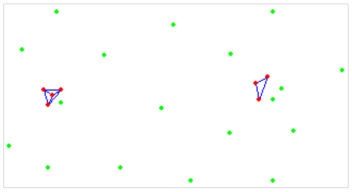
δ=20% β=1: 11 core nodes



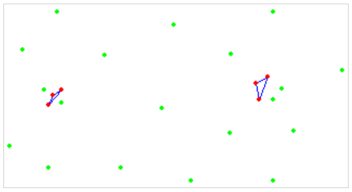
δ=30% β=1: 10 core nodes



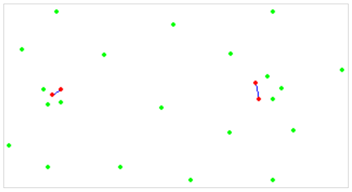
δ=40% β=1: 8 core nodes



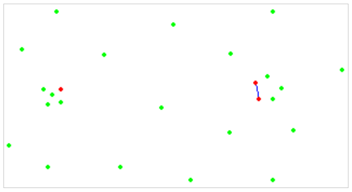
δ=50% β=1: 7 core nodes



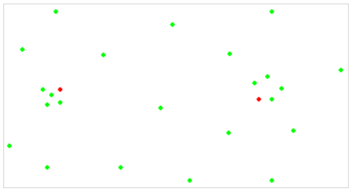
δ=60% β=1: 6 core nodes



δ=70% β=1: 4 core nodes

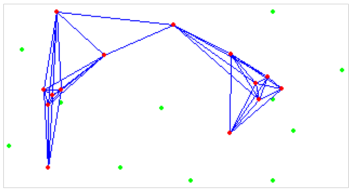


δ=80% β=1: 3 core nodes

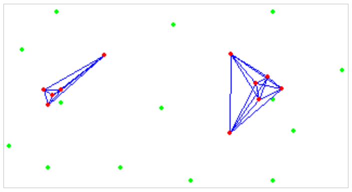


δ=90% β=1: 2 core nodes

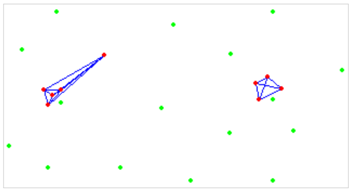
Effects of the parameters



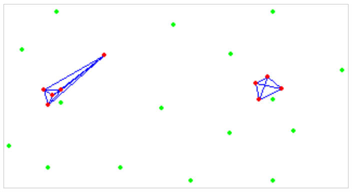
δ=0% β=1: 14 core nodes



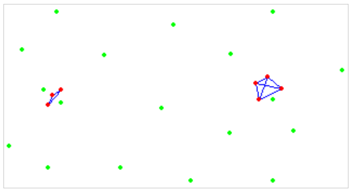
δ=0% β=2: 11 core nodes



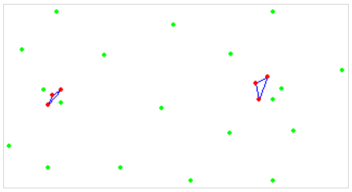
δ=0% β=3: 9 core nodes



δ=0% β=4: 9 core nodes

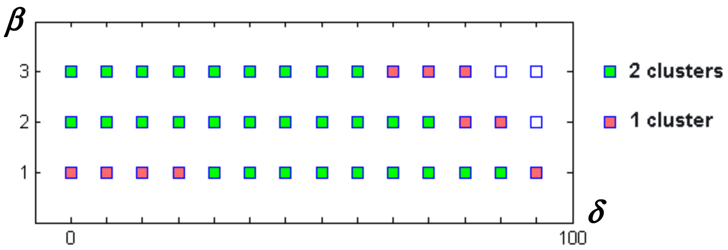


δ=40% β=3: 7 core nodes



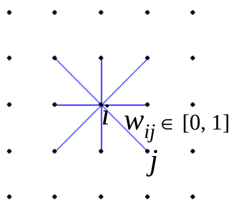
δ=50% β=3: 6 core nodes

Parameter space:



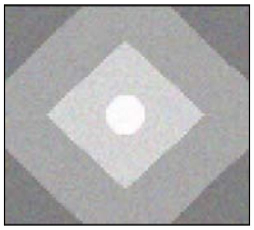
Experiment 1: Image segmentation

- Partition a grayscale image into regions of nearby pixels that have a similar intensity
- Construct a proximity graph from an image: Nodes represent pixels. Edges weights measure the likelihood that two pixels belong to the same segment [J. Shi, J. Malik]

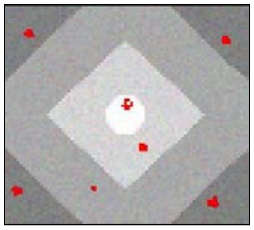


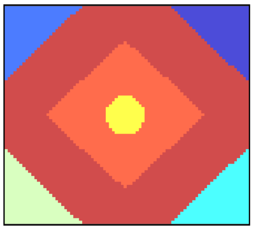
$$w_{ij} = \begin{cases} e^{-\left(\frac{I(i)-I(j)}{\sigma_I}\right)^2 - \left(\frac{d(i,j)}{\sigma_d}\right)^2} & \text{if } d(i,j) < r \\ 0 & \text{otherwise} \end{cases}$$

where $I(x) \in [0, 1]$ $\sigma_I = 0.15$ $\sigma_d = 5$ $r = 6$



$\delta = 96\%$
$\beta = 2$

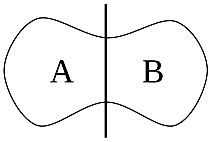




Normalized cuts approach

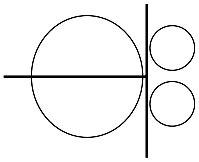
[J. Shi, J. Malik]

- One of spectral clustering approaches: Partition a graph by finding normalized cuts

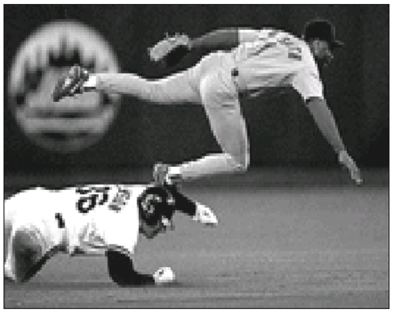


$$Ncut(A,B) = \frac{cut(A,B)}{\sum_{u\in A, v\in V} w_{uv}} + \frac{cut(A,B)}{\sum_{u\in B, v\in V} w_{uv}}$$

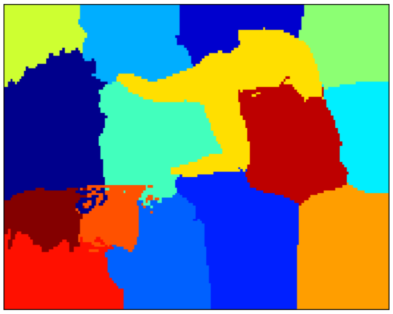
- A NP-hard problem. Approximate a solution by the eigenvector of the second smallest eigenvalue of the normalized Laplacian matrix $I - D^{-1}W$, where D is the diagonal matrix of vertex degrees
- High complexity due to eigenvector computation. The normalizing factors make the criterion favor balanced clusters



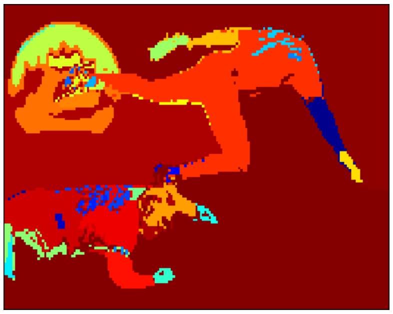
Segmentation examples



[from J. Shi, J. Malik]

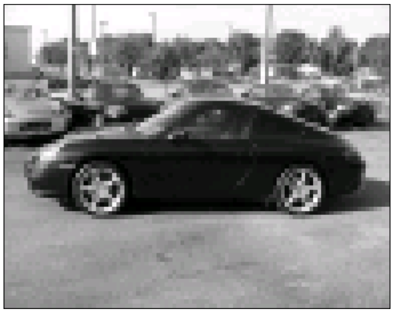


Segmentation by Normalized cuts

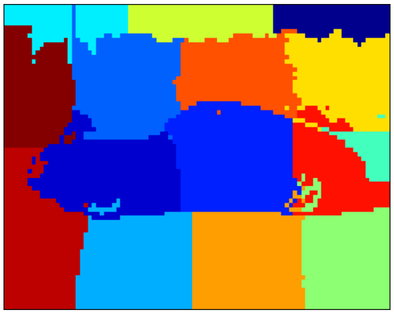


δ= 95.8% β= 2: 34 segments

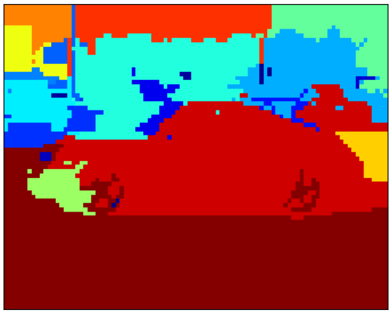
Segmentation examples



[from A. Elgammal]

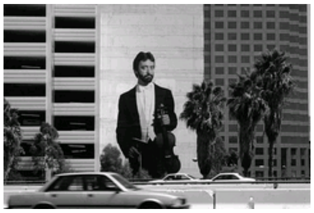


Segmentation by Normalized cuts

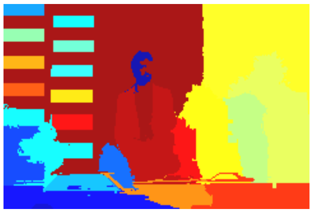


$\delta = 95.5\%$ $\beta = 2$: 22 segments

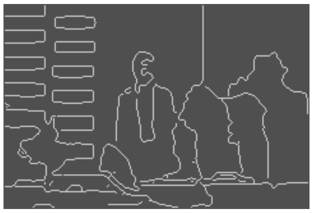
An image from Berkeley segmentation dataset



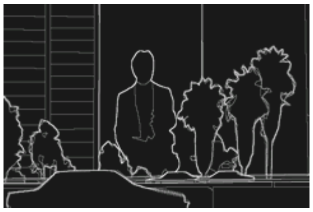
Test image #1



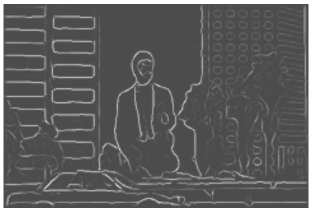
Our segmentation



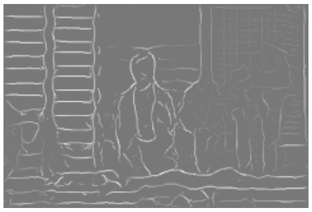
Boundaries derived
from our segmentation
using ‘Canny’ edge detector



Ground truth boundaries
by humans



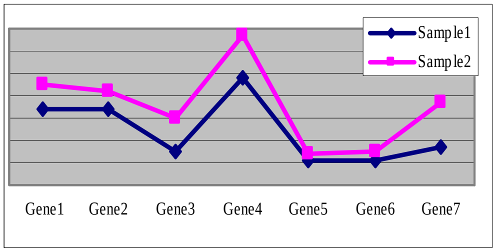
Boundaries detected by
Brightness Gradient method



Boundaries detected by
Texture Gradient method

Experiment 2: Gene expression analysis

- Http://microarray.princeton.edu/oncology/affydata/index.html
- The dataset contains 62 samples including 40 tumor and 22 normal colon tissues
- Each sample consists of a vector of 2000 gene expressions
- Set aside the labels and cluster the samples
- The Pearson correlation coefficient is to measure the similarity between shapes of expression patterns of every pair of samples:



$$r(X,Y) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s_x}\right)\left(\frac{y_i - \bar{y}}{s_y}\right)$$

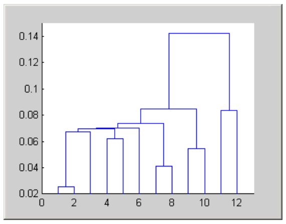
$\bar{x}, \bar{y}$: means of X and Y

s_x, s_y: standard deviations of X and Y

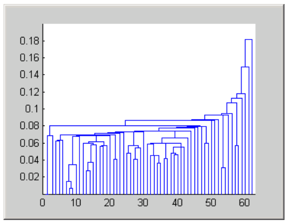
Tissue clustering

Cluster the proximity graph, which is a complete graph of 62 nodes:

- 12 core nodes are identified with $\delta = 50\%$ and $\beta = 2$
- The dendrogram of the core set shows 2 well-separated groups. Cutting it at height 0.1 results in 2 cluster cores



Dendrogram of the core set



Dendrogram of the original data

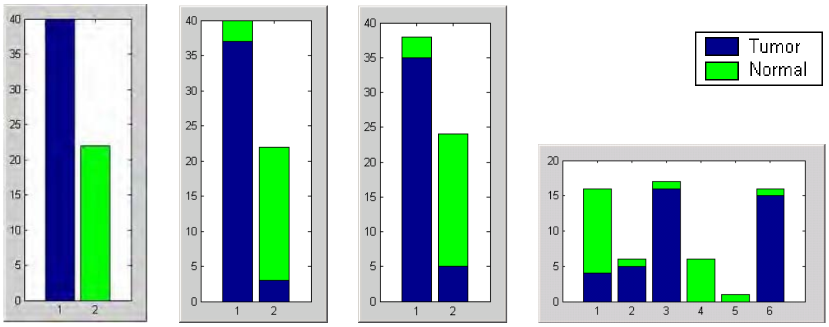
- Expanding 2 cluster cores produces 2 clusters:
  - Cluster 1 has 40 samples: 37 tumor, 3 normal tissues
  - Cluster 2 has 22 samples: 3 tumor, 19 normal tissues

Tissue clustering results

[1] U. Alon et al., Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide array. *Proc. Natl. Acad. Sci. USA*, 1999.

[2] A. Ben-Dor et al., Clustering gene expression patterns. *Journal of Computational Biology*, 1999.

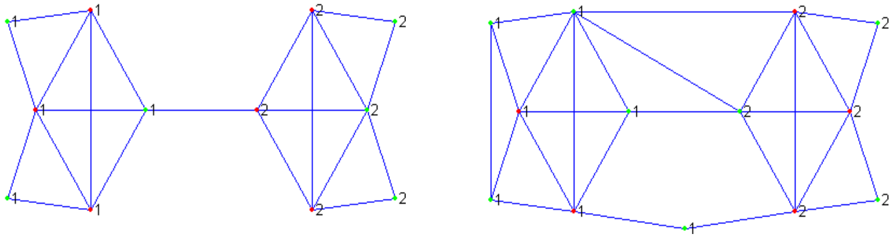
[3] A. Bellaachia et al., A data mining algorithm for gene expression data. *Workshop on Data Mining in Bioinformatics (BIOKDD)*, 2002.



Ideal Coring [1] [2] and [3]

Coring method advantages

- It works on both unweighted and weighted graphs
- Core nodes represent informative data objects
- Ability of manipulating core nodes by humans increases flexibility
- Cluster structures may be revealed by the ranking of core nodes and the confidence degrees of non-core nodes
- Robustness:



Coring method advantages

- Advantages in comparison to the approach based on the maximizer:
  - Has a better time complexity because the time-consuming greedy procedure is run only once - for all parameter settings
  - Can work on highly connected graphs where the maximizer usually covers almost the whole graph
  - Creates no orphan nodes that cause singleton clusters
  - Provides parameters to adjust clustering results and allows human intervention and verification in the course of clustering (steps 2 and 3)

Open issues

- It works best if every cluster has a strong dense core. If some cluster does not have a core or only has a faint core, it may be omitted due to not having representatives in the core set
- Range of parameter settings for good clustering is different for different graph structures. The best parameter range is learnt experimentally
- In case of complete or highly connected graphs, core nodes may be connected in one component, the core set is partitioned by visualization

Future work

- Further experiment and evaluate clustering results using images, gene expression data, or protein sequences
- Perform comparative analysis on different approaches to test the reliability of the method
- Study the possibility of relaxing the dense core assumption or detecting its violations
- Test with new linkage functions such as the consistent triplets to exploit higher level relation of data objects or the linkage function for multipartite graph clustering

Future work

- Analyze parameter space to infer good settings for parameters
- Partition the core set automatically for complete or strongly connected graphs
- New methods of expanding cluster cores for better boundaries between clusters
- Ways to detect cluster structures and recursively apply the method to cluster multi-scale data
- Application to directed graphs

Conclusions

- Clustering methods by quasi-concave set function optimization have been studied. A new graph clustering method is proposed
- The method is simple and fast. Its two control parameters have clear interpretations
- Experiments have been done on synthetic and real datasets with encouraging results
- Further work on open issues, method improvements, experiments, clustering comparison and validation

References

[1] B. Mirkin and I. Muchnik, 2002, Layered clusters of tightness set functions, Applied Mathematics Letters, 15, 147-151

[2] A. Vashist, C. Kulikowski, and I. Muchnik, 2005, Screening for ortholog clusters using multipartite graph clustering by quasi-concave set function optimization

[3] H. Yun, 2005, Triplet-consistency graph clustering

[4] H. Yun, A. Anghelescu and D. Fradkin, Graph clustering codes

[5] *http://www.bioinfo.org.cn/lectures/index-13.html*

[6] *http://www.ics.uci.edu/~eppstein/161/960201.html*

[7] J. Shi and J. Malik, 2000, Normalized cuts and image segmentation, IEEE Trans. Patt. Anal. Mach. Intel., 22, 8, 888-905

[8] U. Alon et al., 1999, Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide array. Proc. Natl. Acad. Sci. USA

[9] A. Ben-Dor et al., 1999, Clustering gene expression patterns. Journal of Computational Biology

[10] A. Bellaachia et al., 2002, E-CAST: A data mining algorithm for gene expression data. Workshop on Data Mining in Bioinformatics (BIOKDD)