Lecture 10. Clustering Graphs and Networked Data

Lecture 10. Clustering Graphs and Networked Data

- Graph and Network Clustering: Basic Concepts
- Graphs, Networks, and Their Representations
- Typical Evaluation Measures
- Approaches for Graph Clustering
- Spectral Clustering
- SCAN: Density-Based Clustering of Networks
- Summary

Session 1: Graph and Network Clustering: Basic Concepts

Graph and Network Clustering: Basic Concepts

- Most real-world data are inter-connected, forming gigantic networks/graphs
- **Homogeneous networks** vs. **heterogeneous networks**
  - **Homogeneous networks**: Vertices and edges are of one type
    - Web search engines, e.g., click through graphs and Web graphs
    - Social networks, friendship networks, coauthor graphs
  - **Heterogeneous networks**: Vertices and edges are of multiple types
    - Two-typed graphs, e.g., customers and products, authors and conferences
    - Multiple-typed graphs, e.g., research networks, medical networks, Freebase
- Clustering vs. graph partitioning vs. community discovery
  - **Clustering** objects into groups, hard/soft, complete/partial, balanced/skewed
  - **Graph partitioning:** Hard, complete, typically balanced
  - **Community discovery:** Can be partial, often interested only in finding the densely connected components and not in the cluster assignment of every vertex

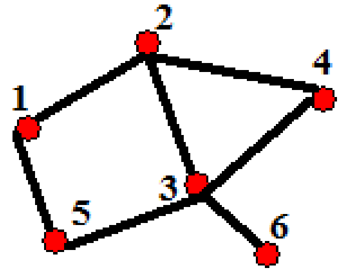
Graph Clustering: Challenges of Finding Good Cuts

- **High computational cost**
  - Many graph cut problems are computationally expensive
  - Need to tradeoff between efficiency/scalability and quality
- **Sophisticated graphs**
  - May involve weights and/or cycles
- **High dimensionality**
  - A graph can have many vertices
  - In a similarity matrix, dimensionality is the number of vertices in the graph
- **Sparsity**
  - A large graph is often sparse, meaning each vertex on average connects to only a small number of other vertices
  - A similarity matrix from a large sparse graph can also be sparse

Session 2: Graphs, Networks, and Their Representations

Graphs, Networks, and Their Representations

- A **network/graph**: $G = (V, E)$, where V: vertices/nodes, E: edges/links
  - E: A subset of $V \times V$, $n = |V|$ (order of G), $m = |E|$ (size of G)
  - **Multi-edge** if there exist more than one edge between the same pair of vertices
  - **Loop** if an edge connects a vertex to itself (i.e., (v_i, v_i))
- **Simple network** if a network has neither self-edges nor multi-edges
- **Adjacency matrix**:
  - $A_{ij} = 1$ if there is an edge between vertices i and j; 0 otherwise
- **Directed graph** (digraph) if each edge has a direction (tail → head)
  - $A_{ij} = 1$ if there is an edge from j to i; 0 otherwise
- **Weighted graph**: If a weight w_{ij} (usually a real number) is associated with each edge v_{ij}



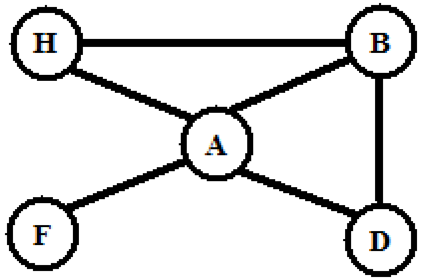
$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Vertex Degree for Undirected & Directed Networks

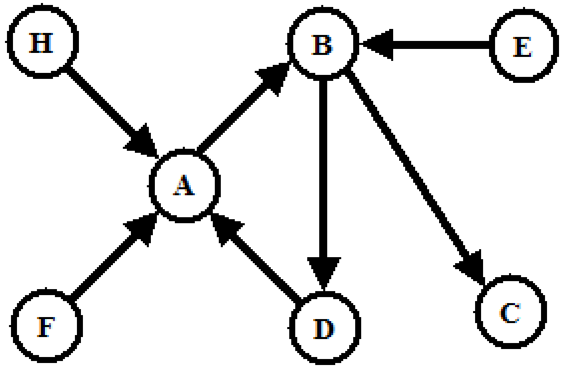
- Let a network $G = (V, E)$
- Undirected Network
  - Degree (or degree centrality) of a vertex: $d(v_i)$

$$d(v_i) = |v_j| \ s.t. \ e_{ij} \in E \wedge e_{ij} = e_{ji}$$

  - # of edges connected to it, e.g., $d(A) = 4$, $d(H) = 2$



- Directed network
  - In-degree of a vertex $d_{in}(v_i)$: $d_{in}(v_i) = |v_j| \ s.t. \ e_{ij} \in E$
    - # of edges pointing to v_i
    - E.g., $d_{in}(A) = 3$, $d_{in}(B) = 2$
  - Out-degree of a vertex $d_{out}(v_i)$: $d_{out}(v_i) = |v_j| \ s.t. \ e_{ji} \in E$
    - # of edges from v_i
    - E.g., $d_{out}(A) = 1$, $d_{out}(B) = 2$

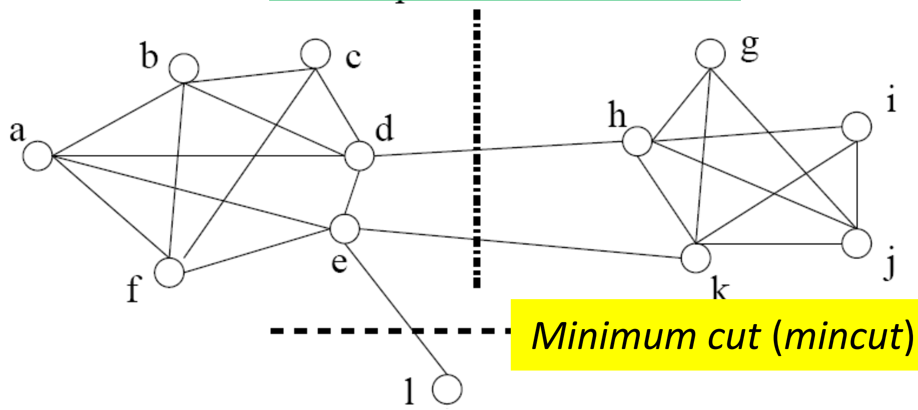


Session 3: Typical Evaluation Measures

Typical Evaluation Measures for Graph Clustering

- Commonly used **measures for graph cutting** To be covered in this session
  - Mincut, ratio cut, normalized cut, conductance, modularity
- Typical **similarity measures across networks**
  - SimRank To be covered in this session
  - Personalized Pagerank

A better cut than *mincut*



Minimum cut (mincut)

- **Minimum cut (*mincut*)**:
  - Given the number k of partitions, choose a partition $C_1, \ldots, C_k$ that minimizes

$$cut(C_1, C_2, \ldots, C_k) = \frac{1}{2}\sum_{i=1}^{k} W(C_i, \overline{C}_i)$$

  where $W(C_i, \overline{C}_i)$ is the sum of the weights of the edges connecting C_i and those in other partitions

  - Mincut can be a poor cut (e.g., cutting one node from the remaining of the graph)

Other Graph Cutting Measures (I)

- Motivation: Make partitions *balanced*
- **Ratio cut**

$$RatioCut(C_1, C_2, \ldots, C_k) = \frac{1}{2}\sum_{i=1}^{k}\frac{W(C_i, \overline{C}_i)}{|C_i|}$$

  - The sum of the edge weights connecting a cluster *C* to the rest of the graph normalized by the size of *C*
- **Normalized cut**

$$NCut(C_1, C_2, \ldots, C_k) = \frac{1}{2}\sum_{i=1}^{k}\frac{W(C_i, \overline{C}_i)}{d(C_i)}$$

  - The sum of the edge weights connecting *C* to the rest of the graph normalized by the total degree of cluster *C*
  - Low normalized cut: Good communities (well connected among themselves and sparsely connected to the rest of the graph)
- **Conductance**

$$Conductance(C) = \frac{\sum_{i \in C, j \in \overline{C}} W(i, j)}{\min\left(\sum_{i \in C} d(i), \sum_{j \in \overline{C}} d(j)\right)}$$

  - Similar to normalized cut
  - The sum of conductance of a partition of a graph into *k* clusters $C_1, \ldots, C_k$ is the sum of the normalized cut or conductance of the individual partitions C_i for $i = 1, \ldots, k$

Other Graph Cutting Measures (II)

- **Modularity**
  - Formula: $$Q = \sum_{i=1}^{k}\left[\frac{W(C_i, C_i)}{e} - \left(\frac{d(C_i)}{2e}\right)^2\right],$$

    where C_is are clusters, e is the number of edges, and $d(C_i)$ represents the total degree of cluster C_i
  - For k clusters, the **modularity** of a clustering assesses the quality of the clustering: The optimal clustering of graphs maximizes the modularity
  - Definition of modularity: The sum of the differences, for each cluster, between fraction of internal edges and fraction of edges that are expected to be inside a random cluster with same total degree
  - Interpretation: The *modularity* of a clustering of a graph is the difference between the fraction of all edges that fall into individual clusters and the fraction that would do so if the graph vertices were randomly connected
- Unfortunately, optimizing any of these *normalized* objective functions is NP-hard

SimRank: Similarity Based on Random Walk and Structural Context

- **Walk** in a graph G between nodes *X* and *Y*: Ordered sequence of vertices, starting at *X* and ending at *Y*, such that there is an edge between every pair of consecutive vertices
- **SimRank:** Structural-context similarity (i.e., based on the similarity of its neighbors)
  - **Neighborhood:** In a directed graph $G = (V, E)$,
    - Individual ***in-neighborhood*** of v: $I(v) = \{u \mid (u, v) \in E\}$
    - Individual ***out-neighborhood*** of v: $O(v) = \{w \mid (v, w) \in E\}$
- **Similarity** defined by SimRank: (where *C* is a constant between 0 and 1)

$$s(u,v) = \frac{C}{|I(u)||I(v)|} \sum_{x \in I(u)} \sum_{y \in I(v)} s(x,y)$$

Initialization: $$s_0(u,v) = \begin{cases} 0 & \text{if } u \neq v \\ 1 & \text{if } u = v \end{cases}$$

  - Then compute s_{i+1} from s_i based on the definition
- It is costly to compute SimRank
  - Many efficient computation methods have been proposed

Session 4: Approaches for Graph Clustering

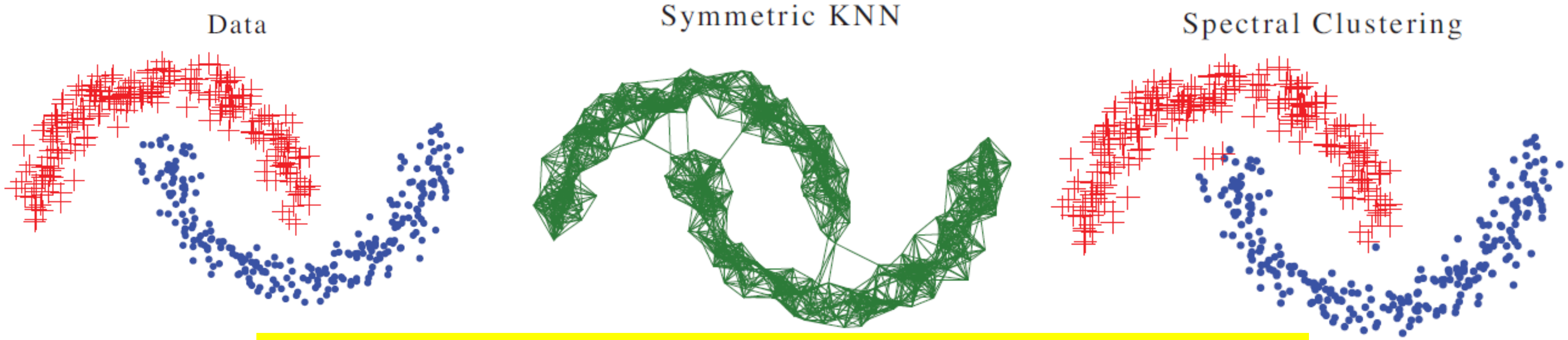
Approaches for Graph Clustering

- **Partition with geometric information** (e.g., Geometric Bisection)
  - Limited to graphs whose geometric info (i.e., meshes) is known
- **Graph growing and greedy algorithms**
  - Ex. Kernigham-Lin (K/L) (1970): Take random partitions and then apply K/L to it
  - K/L: At each iteration, swap pairs of vertices to maximize the gain
- **Agglomerative and divisive clustering**
  - Ex. Newman (2004): Merge small communities if it increases the graph's modularity
- **Spectral clustering** To be covered in the next session
  - One of the most popular clustering methods recently
- **Markov clustering**
  - Iteratively apply *Expand* and *Inflate* on the transition probability matrix
  - *Prune* away the smaller values in each column and renormalize it for next iteration

Session 5: Spectral Clustering

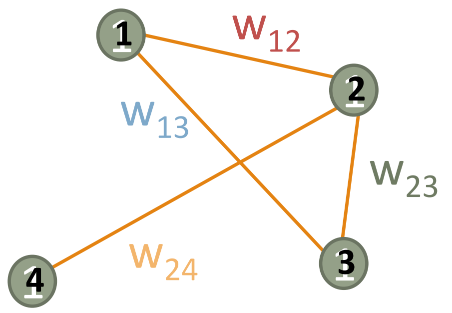
Why Spectral Clustering?

- **Strength of spectral clustering**
  - Makes no assumptions on the shapes of clusters, can handle intertwined spirals, etc.
  - EM or the like require an iterative process to find local minima and multiple restarts
- **Process of spectral clustering**
  - Construct a similarity graph (e.g., KNN graph) for all the data points
  - Embed data points in a low-dimensional space (*spectral embedding*), in which the clusters are more *obvious*, with the use of the eigenvectors of the graph Laplacian
  - A classical clustering algorithm (e.g., *k*-means) is applied to partition the embedding



An illustration of the steps of spectral clustering

Matrix Representations of a Graph



- Adjacency Matrix: n x n symmetric matrix

$$A_{ij} = \begin{cases} w_{ij} & : \text{weight of edge } (i,j) \\ 0 & : \text{if no edge between } i,j \end{cases}$$

A =

	1	2	3	4
1	0	w_{12}	w_{13}	0
2	w_{12}	0	w_{23}	w_{24}
3	w_{13}	w_{23}	0	0
4	0	w_{24}	0	0

- The Laplacian

$$L = D - A$$

$$d_i = \sum_{\{j|(i,j)\in E\}} w_{ij}$$

where D is the diagonal matrix of degrees

$d_1 = w_{12} + w_{13}$
$d_2 = w_{12} + w_{23} + w_{24}$
$d_3 = w_{13} + w_{23}$
$d_4 = w_{24}$

$$L_{ij} = \begin{cases} d_i & : \text{if } i = j \\ -w_{ij} & : \text{if } (i,j) \text{ is an edge} \\ 0 & : \text{if no edge between } i,j \end{cases}$$

L =

	1	2	3	4
1	d_1	$-w_{12}$	$-w_{13}$	0
2	$-w_{12}$	d_2	$-w_{23}$	$-w_{24}$
3	$-w_{13}$	$-w_{23}$	d_3	0
4	0	$-w_{24}$	0	d_4

Eigenvalue and Eigenvector of Graph Adjacency Matrix

- For a matrix **A**, λ is an *eigenvalue* of **A** if for some vector ***v***,

$$A\nu = \lambda\nu$$

v is the *eigenvector* of **A** corresponding to λ

- For a graph G with n nodes, its adjacency matrix has n eigenvalues $\{\lambda'_1, \lambda'_2, \ldots, \lambda'_n\}$ where $\lambda'_1 \geq \lambda'_2 \geq \ldots \geq \lambda'_n$, and n corresponding eigenvectors $\{\mathbf{v}'_1, \mathbf{v}'_2, \ldots, \mathbf{v}'_n\}$

Spectrum of a graph:
$\lambda'_1 \geq \lambda'_2 \geq \ldots \geq \lambda'_n$

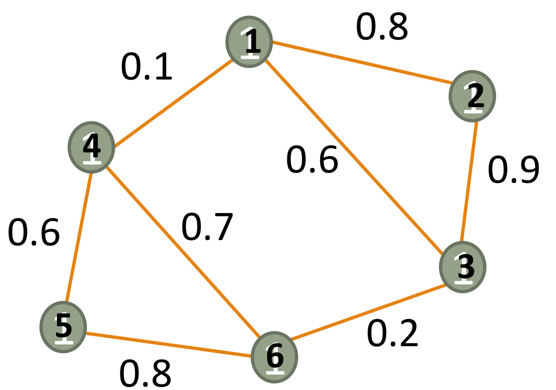


Figure 1. Graph G with adjacency matrix **A**

A =

	1	2	3	4	5	6
1	0	0.8	0.6	0.1	0	0
2	0.8	0	0.9	0	0	0
3	0.6	0.9	0	0	0	0.2
4	0.1	0	0	0	0.6	0.7
5	0	0	0	0.6	0	0.8
6	0	0	0.2	0.7	0.8	0

$\lambda'_1 = 1.584$	$\boldsymbol{v}'_1$ = [-0.48, -0.53, -0.52, -0.26, -0.25, -0.30]
$\lambda'_2 = 1.355$	$\boldsymbol{v}'_2$ = [0.25, 0.30, 0.24, -0.48, -0.52, -0.52]
$\lambda'_3 = -0.482$	$\boldsymbol{v}'_3$ = [-0.57, 0.06, 0.48, -0.56, 0.19, 0.31]
$\lambda'_4 = -0.642$	$\boldsymbol{v}'_4$ = [-0.37, -0.03, 0.36, 0.53, -0.66, 0.13]
$\lambda'_5 = -0.832$	$\boldsymbol{v}'_5$ = [-0.46, 0.44, 0.00, 0.30, 0.37, -0.61]
$\lambda'_6 = -0.993$	$\boldsymbol{v}'_6$ = [-0.17, 0.65, -0.56, -0.11, -0.25, 0.39]

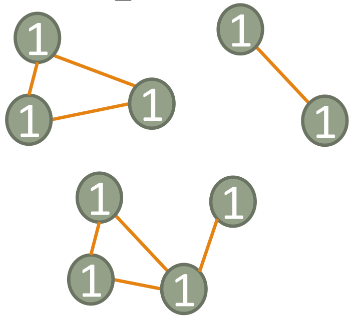
Eigenvalues and eigenvectors of A

Eigenvalue and Eigenvector of Graph Laplacian

- The graph Laplacian of G, L_G, also has
  - eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ where $0 = \lambda_1 \le \lambda_2 \le \ldots \le \lambda_n$, *and*
  - eigenvectors $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$

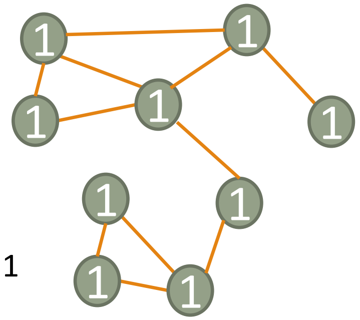
Spectrum of the Laplacian:
$0 = \lambda_1 \le \lambda_2 \le \ldots \le \lambda_n$

- Eigenvalues reveal global graph properties not apparent from edge structure
  - If 0 is the eigenvalue of L with *k* different eigenvectors, i.e., $0 = \lambda_1 = \lambda_2 \ldots = \lambda_k$, then G has *k* connected components
  - If the graph is connected, $\lambda_2 > 0$ and λ_2 is the **algebraic connectivity** of G
    - The greater λ_2, the more connected G is

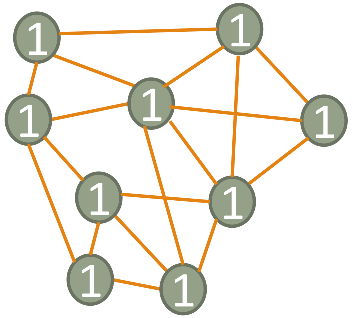


G

L_G has $0 = \lambda_1 = \lambda_2 = \lambda_3$ and $\lambda_4 > 0$



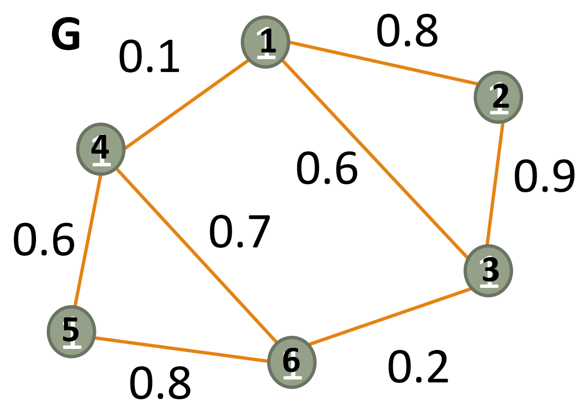
G_1



G_2

Both L_{G1} and L_{G2} have $0 = \lambda_1$ and $\lambda_2 > 0$. $\lambda_2(L_{G1}) < \lambda_2(L_{G2})$

Bi-Partitioning via Spectral Methods



$\lambda_2(L_G) = 0.189$

- To find the bipartition, we take the second eigenvector of the Laplacian, $\boldsymbol{v}_2$, corresponding to λ_2, the algebraic connectivity of G
  - The smaller λ_2, the better quality of the partitioning
- For each node i in G, assign it the value $\boldsymbol{v}_2(i)$ e.g., $\boldsymbol{v}_2(1) = 0.41$ in G
- To find clusters C_1 and C_2, assign nodes with $\boldsymbol{v}_2(i) > 0$ to C_1 and $\boldsymbol{v}_2(i) < 0$ to C_2

$C_1 = \{ i \mid \mathbf{v}_2(i) > 0\}$,
$C_2 = \{ i \mid \mathbf{v}_2(i) < 0\}$

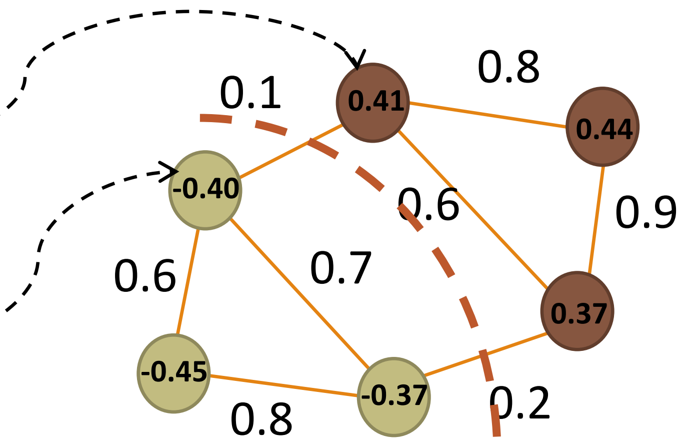
$L_G =$

	1	2	3	4	5	6
1	1.5	-0.8	-0.6	-0.1	0	0
2	-0.8	1.7	-0.9	0	0	0
3	-0.6	-0.9	1.7	0	0	-0.2
4	-0.1	0	0	1.4	-0.6	-0.7
5	0	0	0	-0.6	1.4	-0.8
6	0	0	-0.2	-0.7	-0.8	1.7

Laplacian of ***G***

	$\mathbf{v}_2$
1	0.41
2	0.44
3	0.37
4	-0.40
5	-0.45
6	-0.37

Second eigenvector of L_G



$C_1 = \{1, 2, 3\}$
$C_2 = \{4, 5, 6\}$

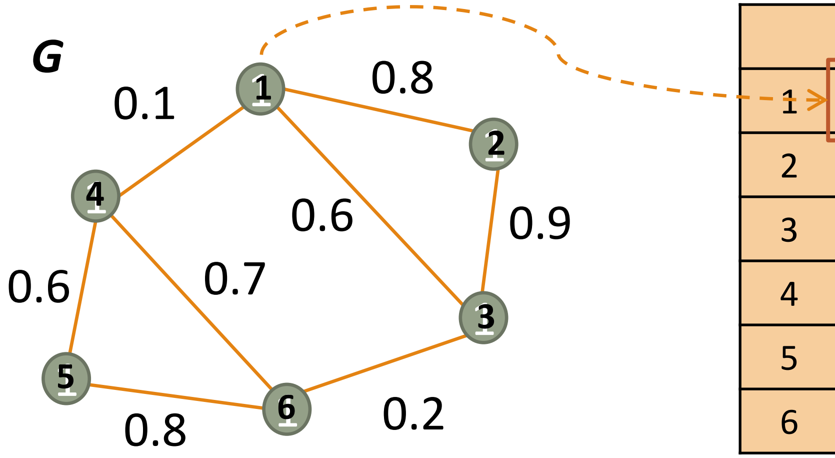
Nodes in ***G*** mapped onto $\boldsymbol{v}_2$
Bipartition of G based $\boldsymbol{v}_2$

Extension to *k* partitions

- **The Ng-Jordan-Weiss (NJW) Algorithm** (2002) $L_{norm} = D^{-1/2} L D^{-1/2}$
  - Compute the first k eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ of L_{norm}, the normalized Laplacian
  - Let $U \in R^{n \times k}$ be the matrix containing the vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ as columns
  - For $i = 1, \ldots, n$, take the i-th row of U as its feature vector after normalizing to norm 1
  - Cluster the points with k-means into k clusters $C_1, \ldots, C_k$
  - Commonly used as a dimension reduction technique for clustering

	1	2	3	4	5	6
1	1.0	-0.5	-0.4	-0.1	0	0
2	-0.5	1.0	-0.5	0	0	0
3	-0.4	-0.5	1.0	0	0	-0.1
4	-0.1	0	0	1.0	-0.4	-0.5
5	0	0	0	-0.4	1.0	-0.5
6	0	0	-0.1	-0.5	-0.5	1.0

$L_{norm}(G)$



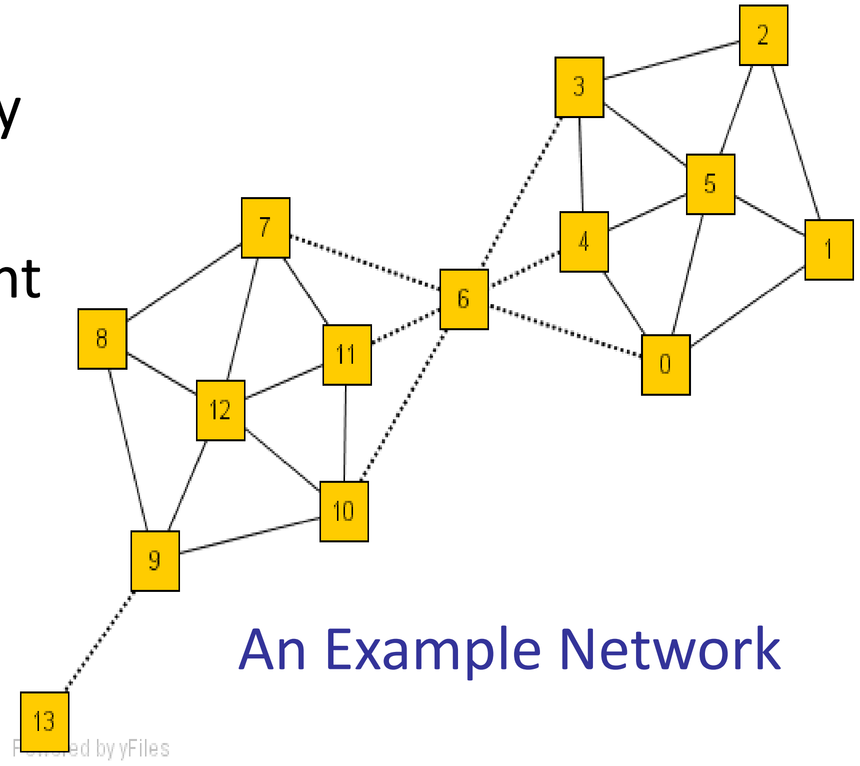
	$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$
1	$v_1(1)$	$v_2(1)$	$v_3(1)$
2	$v_1(2)$	$v_2(2)$	$v_3(2)$
3	$v_1(3)$	$v_2(3)$	$v_3(3)$
4	$v_1(4)$	$v_2(4)$	$v_3(4)$
5	$v_1(5)$	$v_2(5)$	$v_3(5)$
6	$v_1(6)$	$v_2(6)$	$v_3(6)$

U for $k = 3$

Session 6: SCAN: Density-Based Clustering of Networks

Clustering and Social Network Analysis

- Cliques, hubs, and outliers
  - Individuals in a tight social group, or clique, know many of the same people, regardless of the size of the group
  - Individuals who are hubs know many people in different groups but belong to no single group. Politicians, for example bridge multiple groups
  - Individuals who are outliers reside at the margins of society. Hermits, for example, know few people and belong to no group

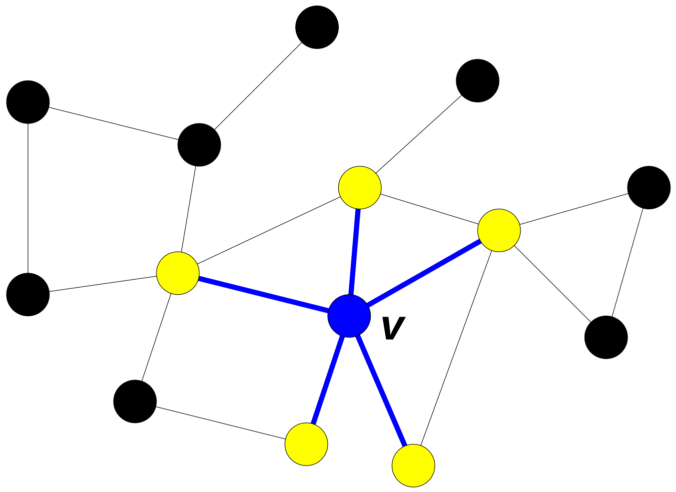


An Example Network

- Application of cluster analysis
  - Given information about who associates with whom
    - Can we identify clusters of individuals with common interests or special relationships (families, cliques, or terrorist cells)?

SCAN: Density-Based Clustering of Networks

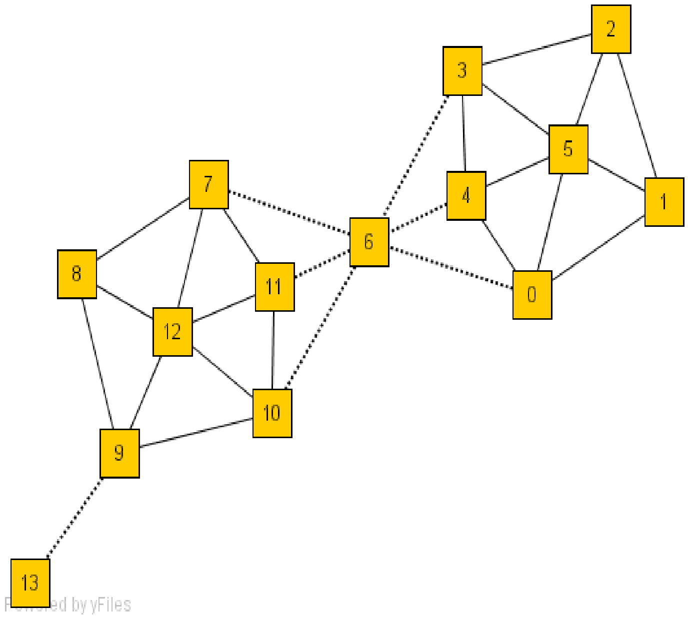
- The **neighborhood** of a vertex v: $\Gamma(\nu)$
  - Given a vertex v, $\Gamma(\nu)$ is defined as ν and the immediate neighborhood of *v* (e.g., the set of people that an individual knows)



- **Similarity** between two vertices *v* and *w*: *σ*(*v, w*)

$$\sigma(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

- The desired features tend to be captured by a measure called ***Structural Similarity***
- Structural similarity is large for members of a clique and small for hubs and outliers



Structural Connectivity (Similar to DBSCAN)

- Structural similarity is defined similarly as DBSCAN (KDD'06)

  Please check *Lecture 5: Density-Based Clustering* for details

  - ε-Neighborhood: $N_{\varepsilon}(v) = \{w \in \Gamma(v) \mid \sigma(v,w) \geq \varepsilon\}$
  - Core: $CORE_{\varepsilon,\mu}(v) \Leftrightarrow |N_{\varepsilon}(v)| \geq \mu$
- Direct structure reachable:

$$DirReach_{\varepsilon,\mu}(v,w) \Leftrightarrow CORE_{\varepsilon,\mu}(v) \wedge w \in N_{\varepsilon}(v)$$

- Structure reachable (REACH(*u*, *v*))
  - Transitive closure of direct structure reachability
- Structure connected:

$$CONNECT_{\varepsilon,\mu}(v,w) \Leftrightarrow \exists u \in V : REACH_{\varepsilon,\mu}(u,v) \wedge REACH_{\varepsilon,\mu}(u,w)$$

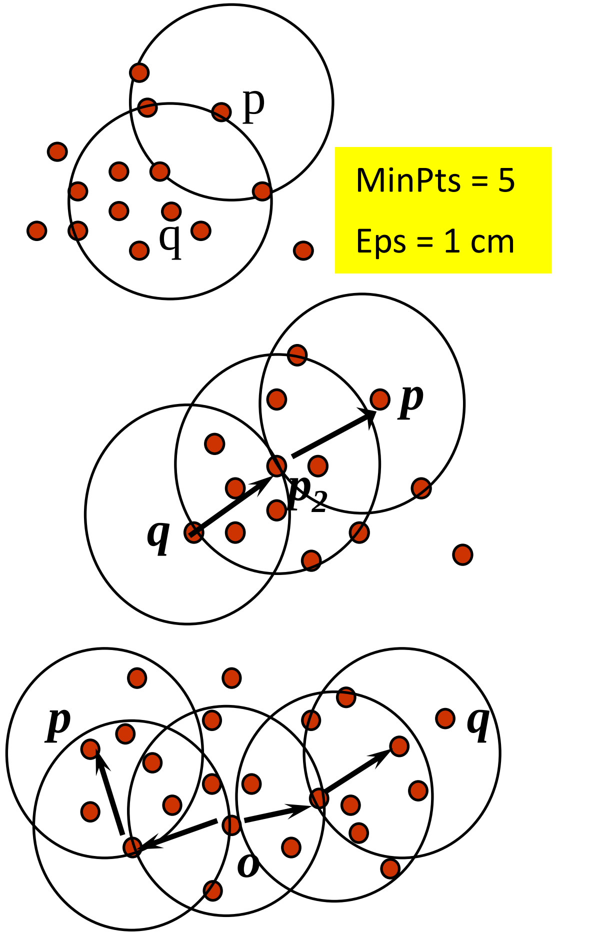
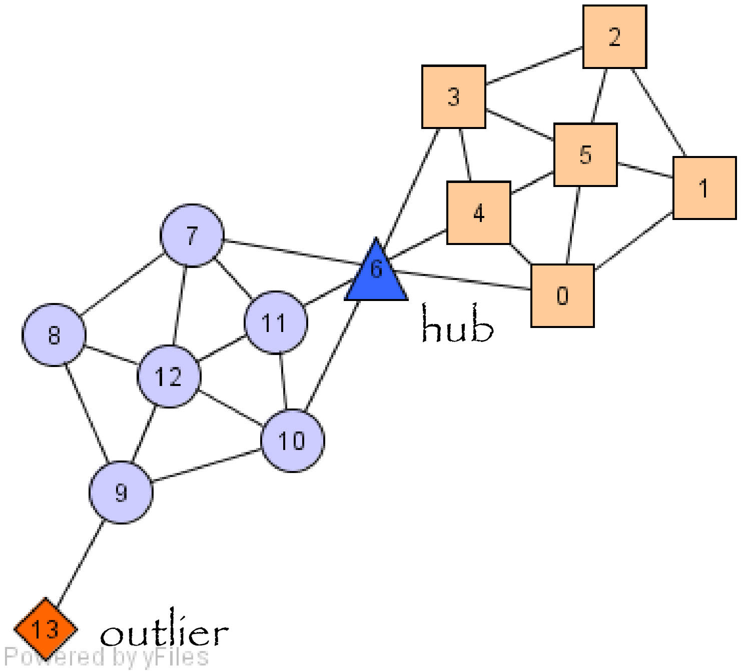


Illustration of the concepts of density-based clustering

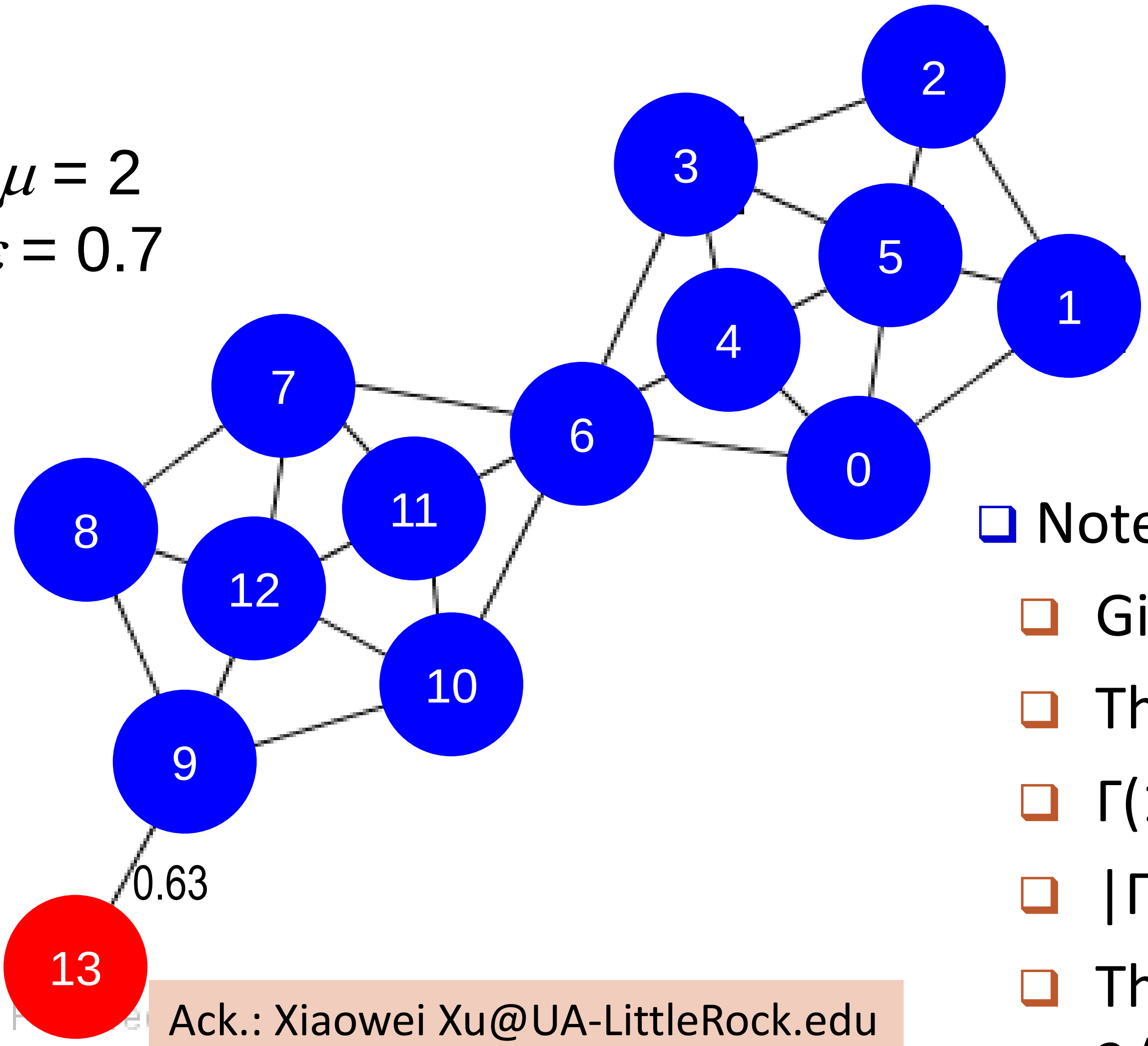
Structure-Connected Clusters

- Structure-connected cluster C
  - Connectivity: $\forall v, w \in C : CONNECT_{\varepsilon,\mu}(v, w)$
  - Maximality: $\forall v, w \in V : v \in C \wedge REACH_{\varepsilon,\mu}(v, w) \Rightarrow w \in C$
- Hubs:
  - Do not belong to any cluster
  - Bridge to many clusters
- Outliers:
  - Do not belong to any cluster
  - Connect to fewer clusters



The Execution of the Scan Algorithm (I)

$\mu = 2$
$\varepsilon = 0.7$

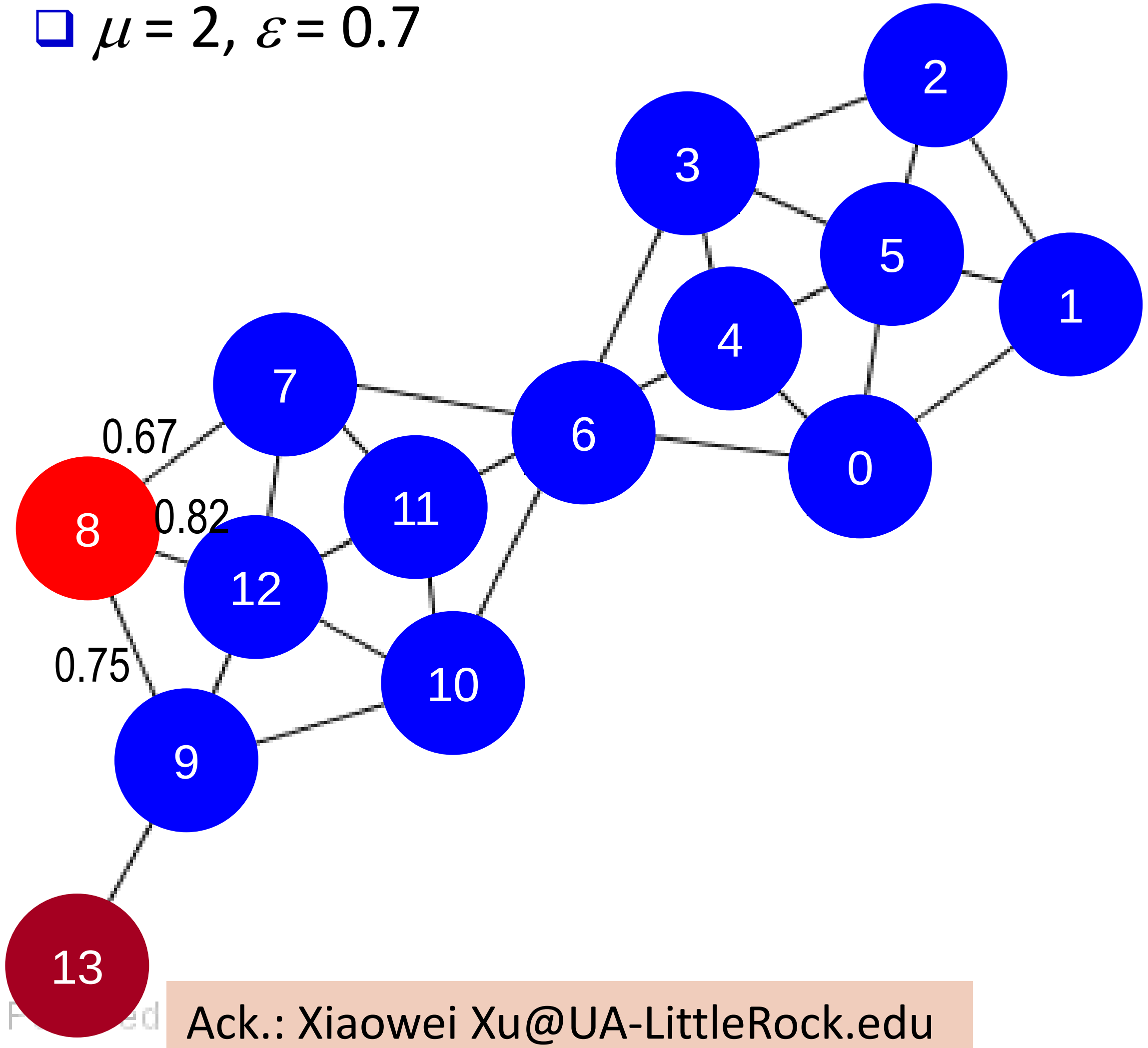


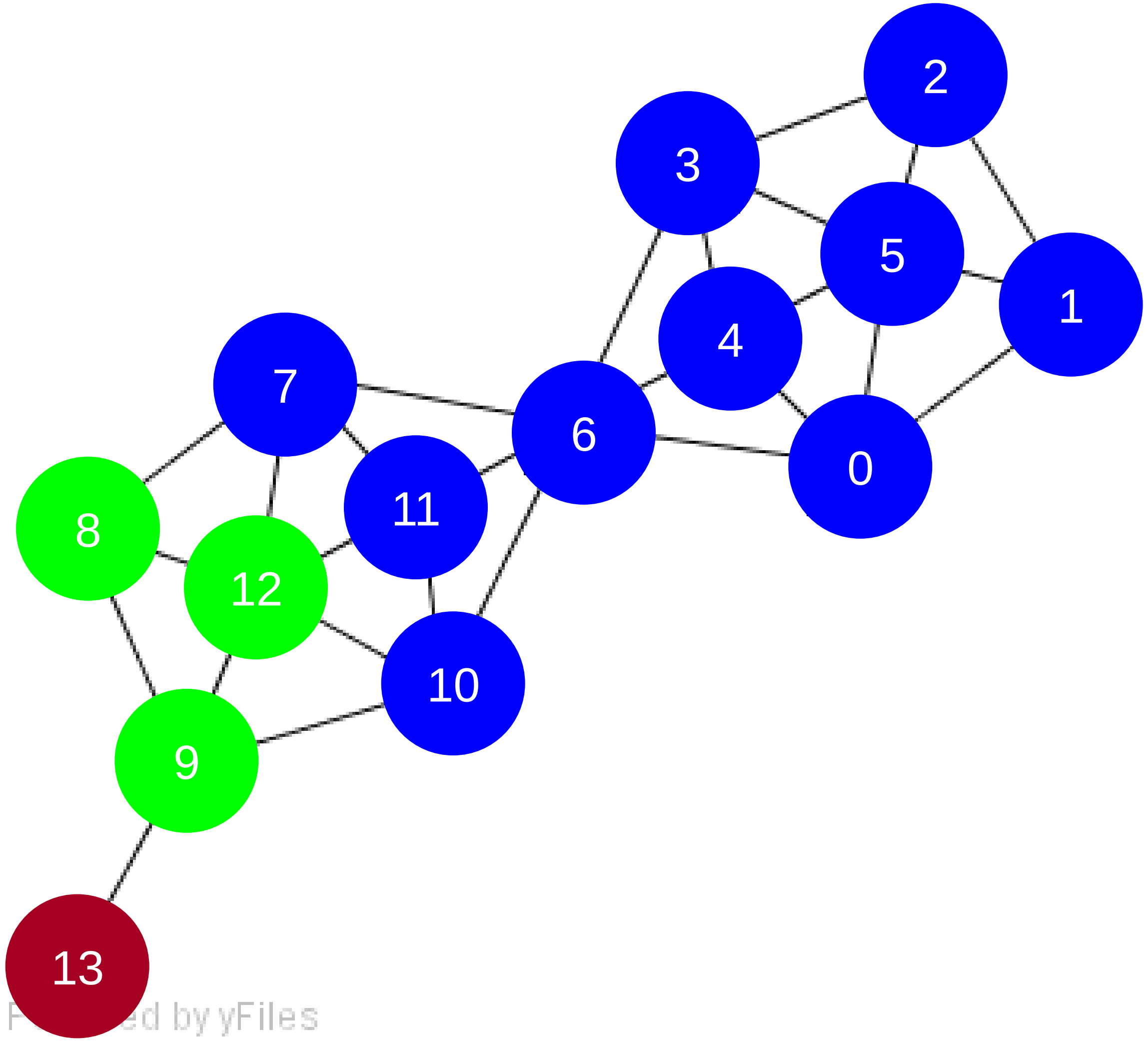
Ack.: Xiaowei Xu@UA-LittleRock.edu

- Note: How is σ = 0.63 derived?
  - Given Γ(13) = {9, 13}, Γ(9) = {9, 13, 8, 12, 10}
  - That is, |Γ(13)| = 2, |Γ(9)| = 5
  - Γ(13) ∩ Γ(9)| = {9, 13}   // set intersection
  - |Γ(13) ∩ Γ(9)| = 2
  - Thus |Γ(13) ∩ Γ(9)|/√(|Γ(13)||Γ(9)|) = 2/√(2*5) = 2/√10 = 2/3.162 = 0.63

The Execution of the Scan Algorithm (II)

- $\mu = 2$, $\varepsilon = 0.7$

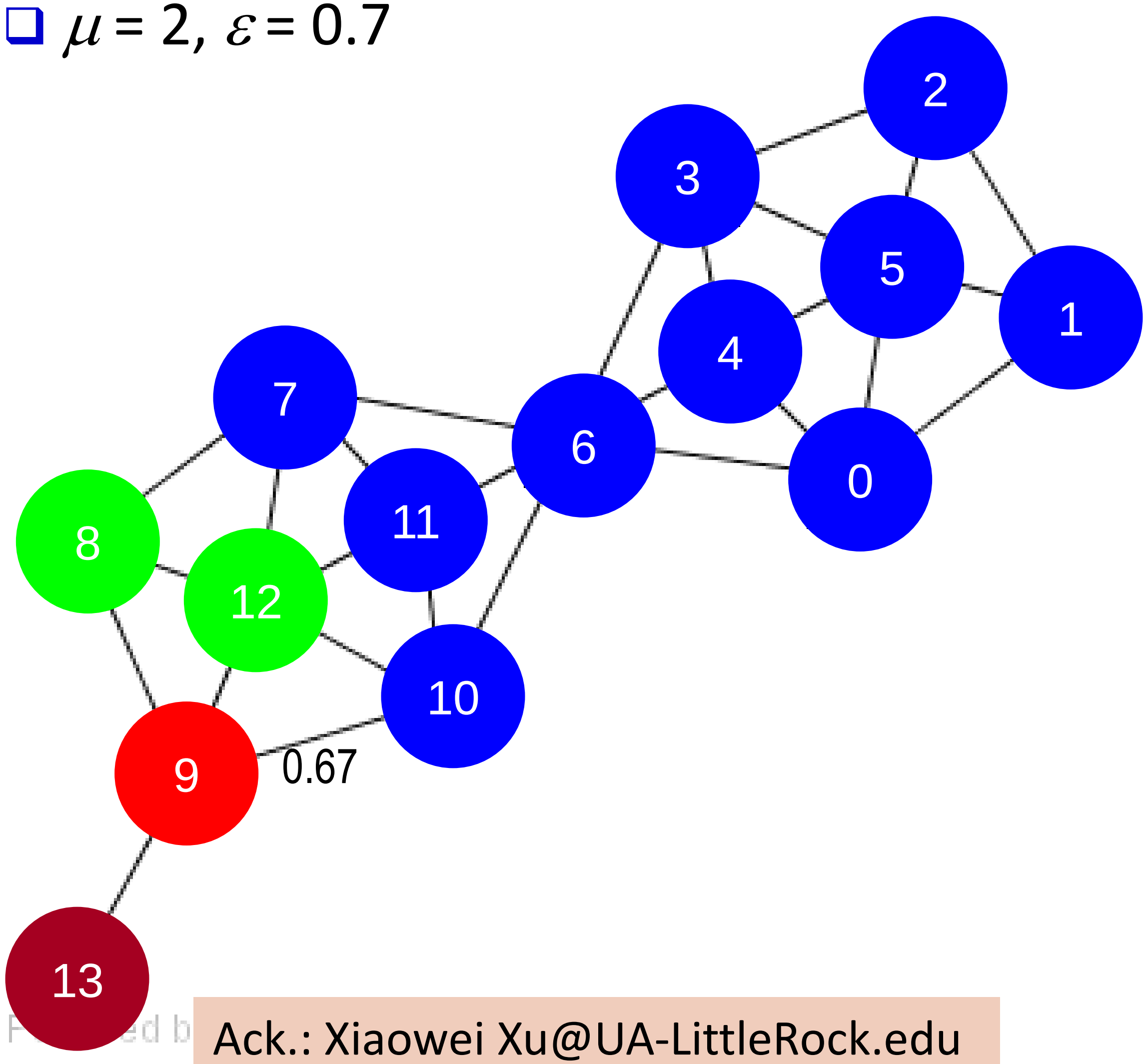


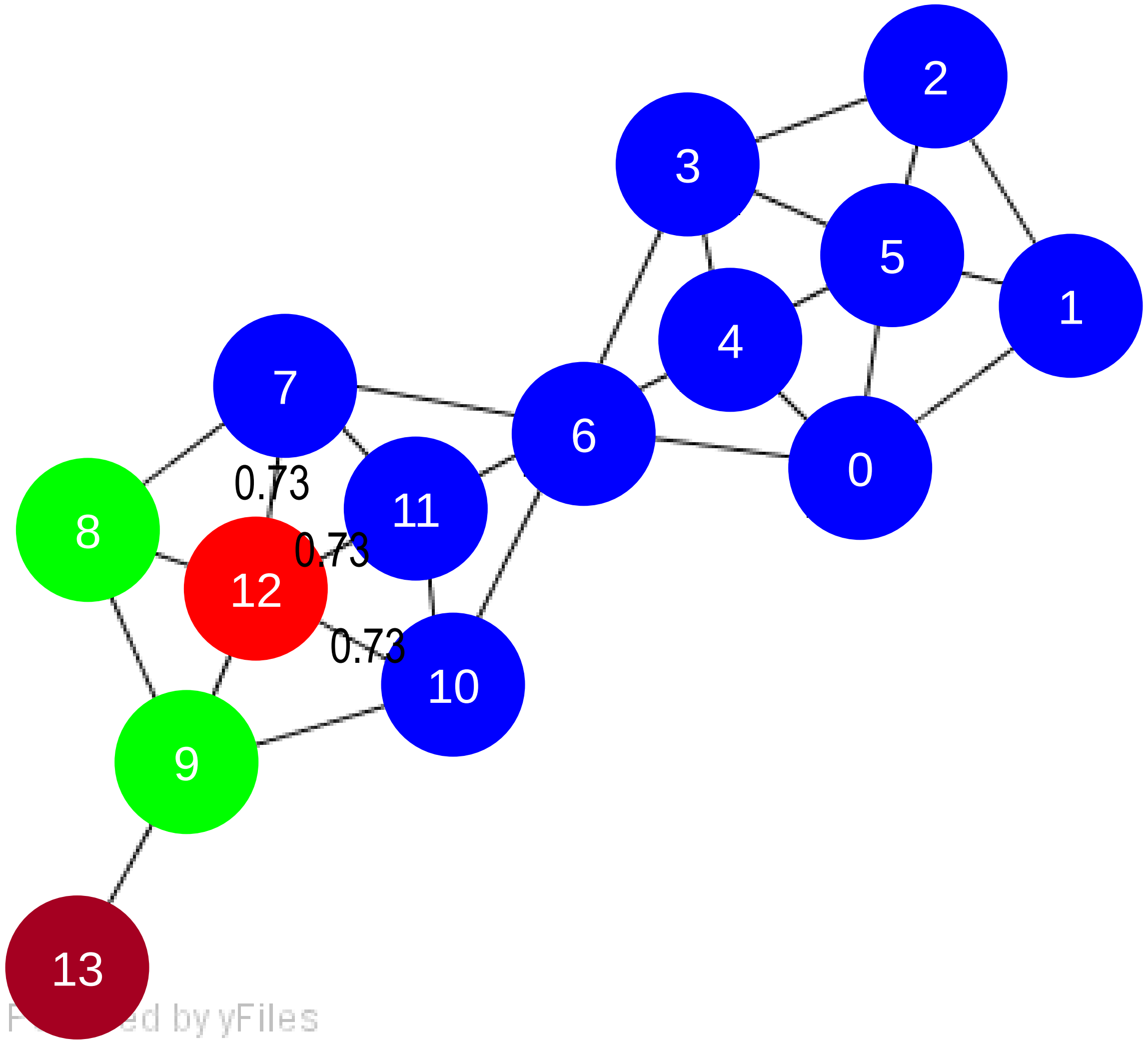


Ack.: Xiaowei Xu@UA-LittleRock.edu

The Execution of the Scan Algorithm (III)

- $\mu = 2$, $\varepsilon = 0.7$

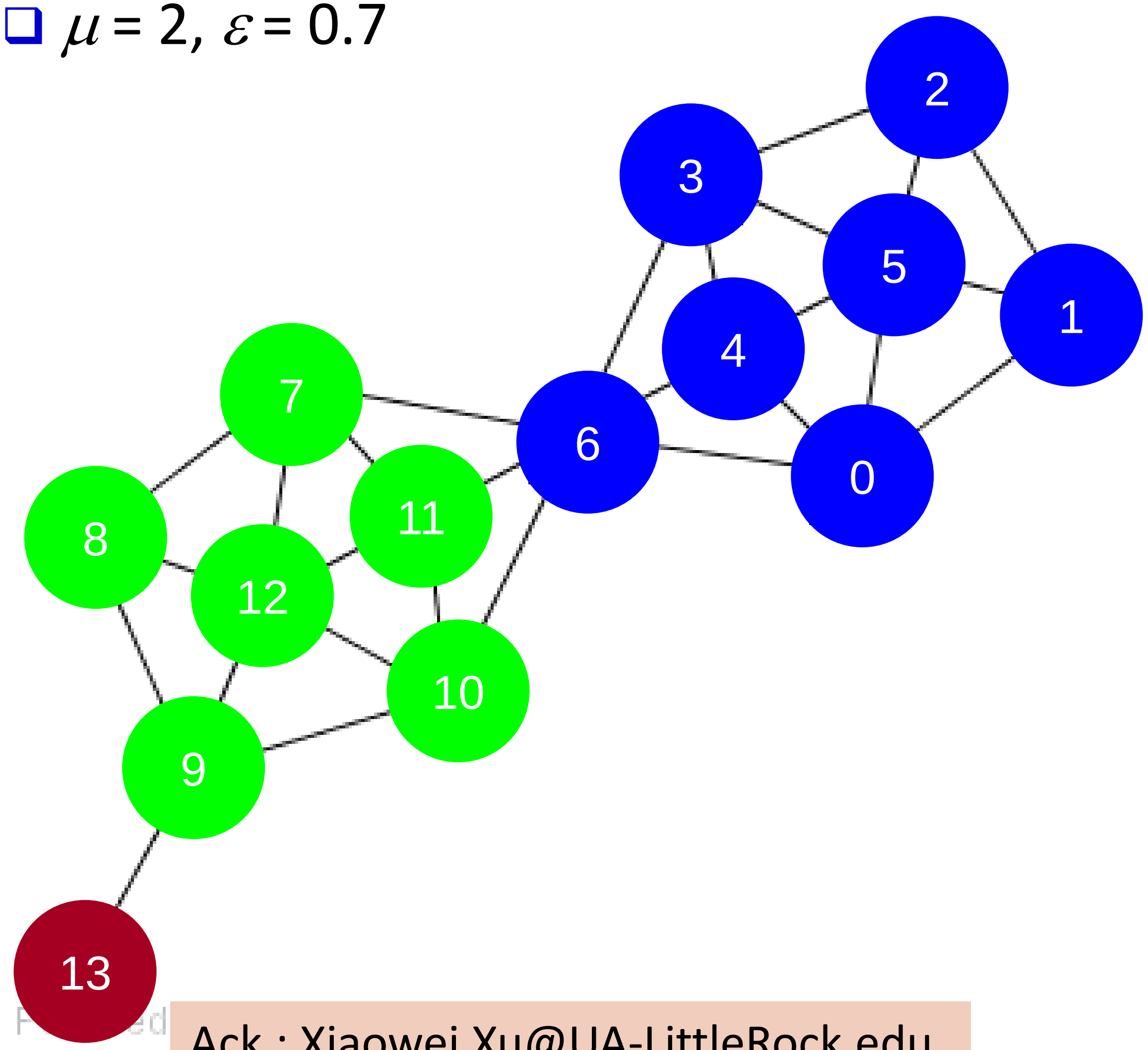


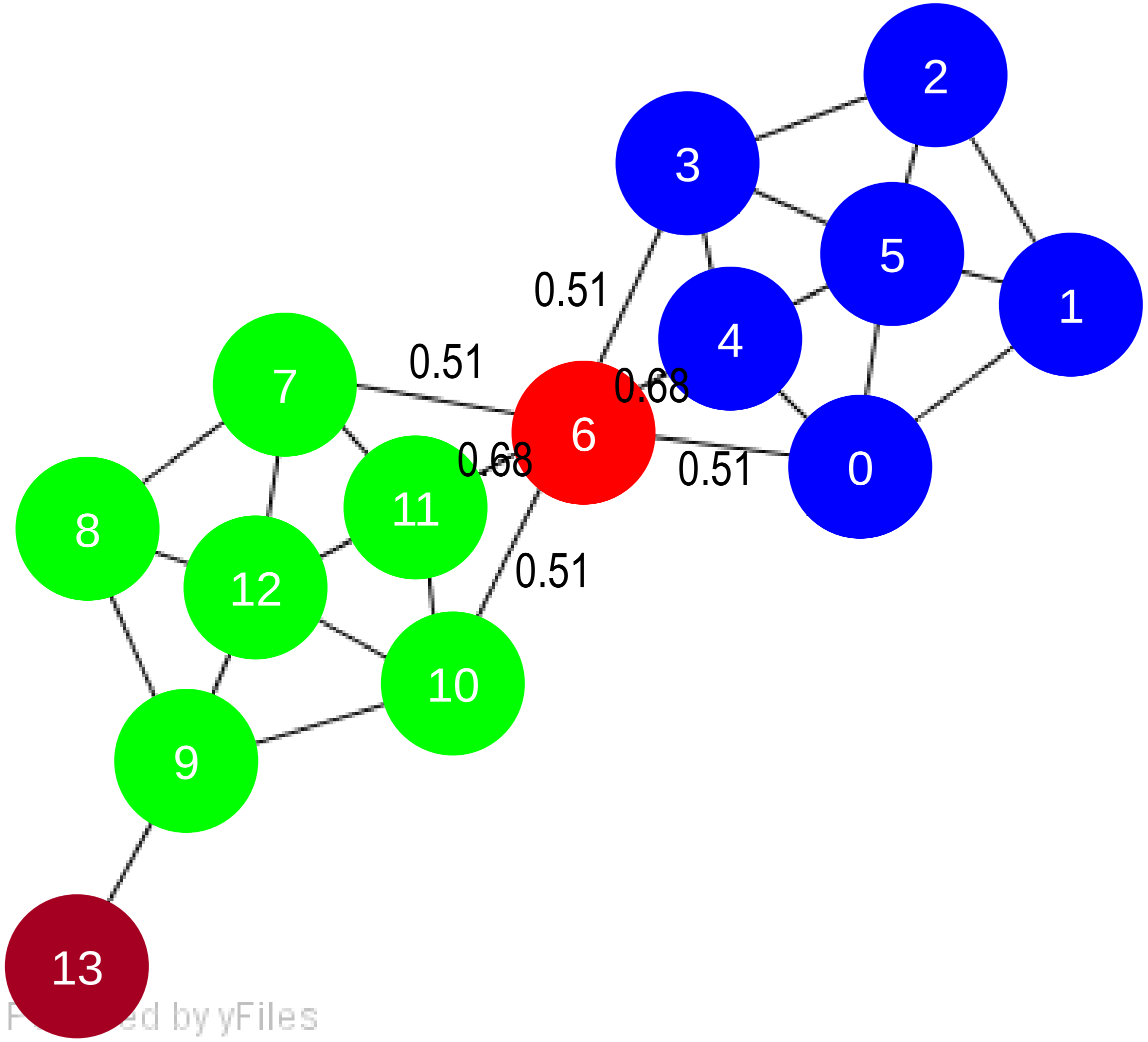


Ack.: Xiaowei Xu@UA-LittleRock.edu

The Execution of the Scan Algorithm (IV)

- $\mu = 2$, $\varepsilon = 0.7$

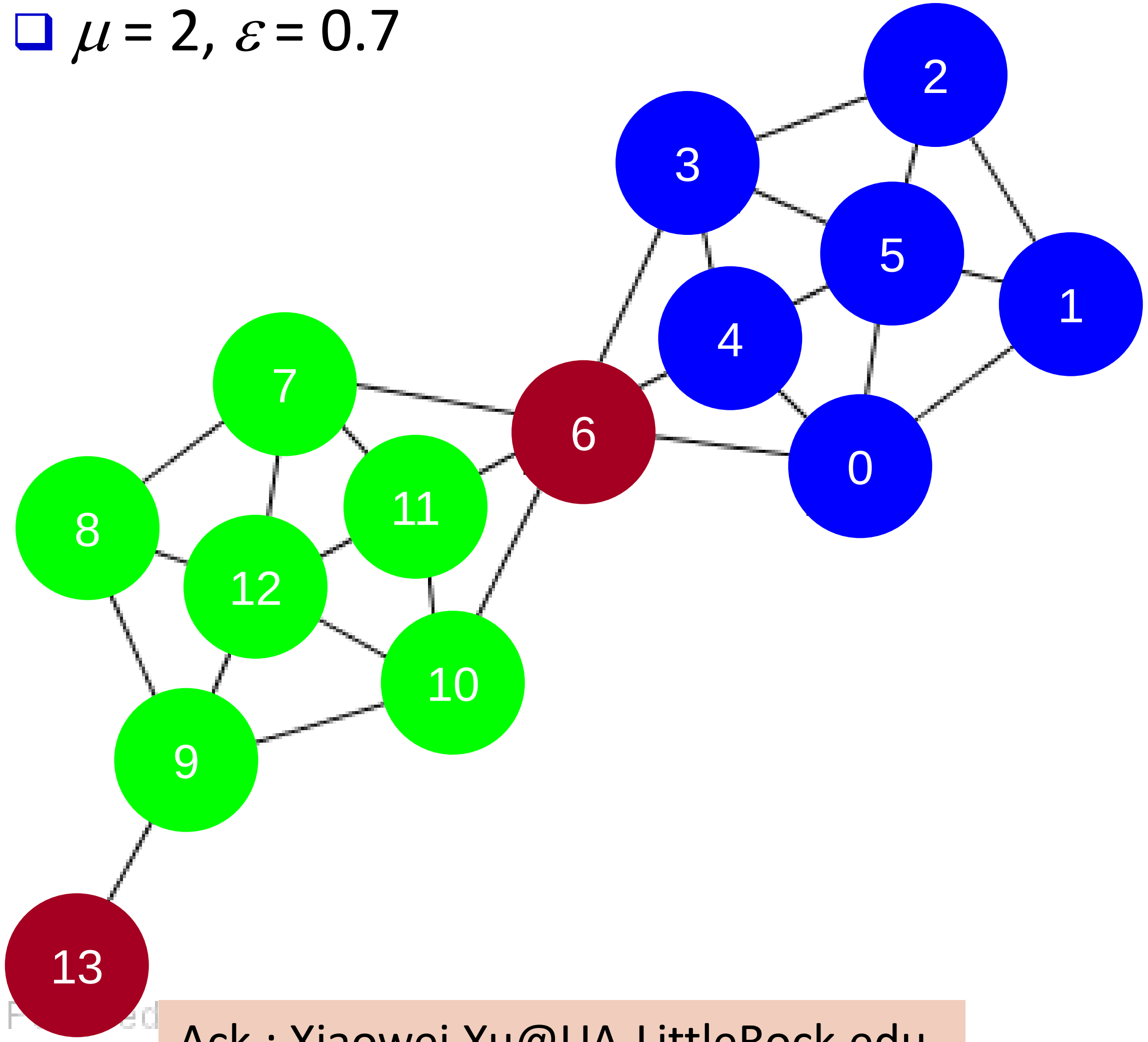




Ack.: Xiaowei Xu@UA-LittleRock.edu

The Execution of the Scan Algorithm (V)

- $\mu = 2$, $\varepsilon = 0.7$

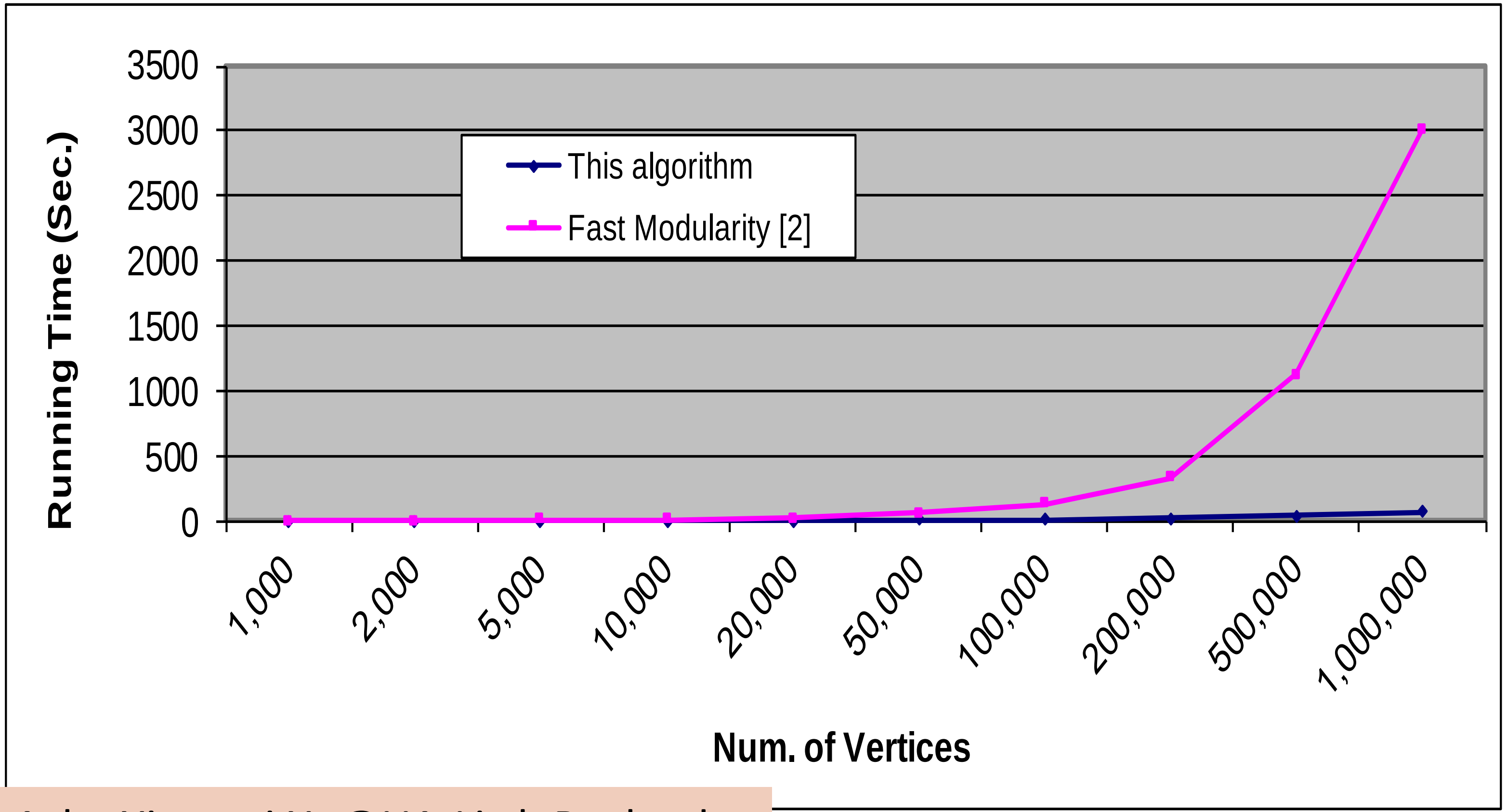


Ack.: Xiaowei Xu@UA-LittleRock.edu

- Continuing execution will group vertices {0, 1, 2, 3, 4, 5} into one cluster
- It finally generates two clusters, one hub vertex {6} and one outlier vertex {13}
- Experiments on several real-world data sets generate interesting clusters (see Xu et al. KDD 2007)
- One major issue: How to set up desirable parameters, μ and ε?
  - Several follow-up studies to handle the problem

Efficiency of the Scan Algorithm

- Computational complexity
  - Running time = $O(|E|)$
  - For sparse networks = $O(|V|)$



- The comparison algorithm:
  - [2] A. Clauset, M. E. J. Newman, & C. Moore, *Phys. Rev.* (2004)

Ack.: Xiaowei Xu@UA-LittleRock.edu

Session 7: Summary

Summary: Clustering Graphs and Networked Data

- Graph and Network Clustering: Basic Concepts
- Graphs, Networks, and Their Representations
- Typical Evaluation Measures
- Approaches for Graph Clustering
- Spectral Clustering
- SCAN: Density-Based Clustering of Networks
- Summary

Recommended Readings

- S. Arora, S. Rao, and U. Vazirani. Expander Flows, Geometric Embeddings and Graph Partitioning. *J. ACM*, 56:5:1–5:37, 2009
- G. Jeh and J. Widom. SimRank: A Measure of Structural-Context Similarity. *KDD'02*
- U. Luxburg. A Tutorial on Spectral Clustering. *Statistics and Computing*, 17, 2007
- A. Y. Ng, M. I. Jordan, and Y. Weiss. On Spectral Clustering: Analysis and an Algorithm. NIPS'01
- S. E. Schaeffer. Graph Clustering. *Computer Science Review*, 1:27–64, 2007
- X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. SCAN: A Structural Clustering Algorithm for Networks. *KDD'07*
- M. J. Zaki and W. Meira, Jr.. Data Mining and Analysis: Fundamental Concepts and Algorithms. Cambridge University Press, 2014
- S. Parthasarathy and S. M. Faisal. Network Clustering, in C. Aggarwal and C. K. Reddy (eds.), Data Clustering: Algorithms and Applications (Chapter 17) . CRC Press, 2014