CS6220: DATA MINING TECHNIQUES

Mining Graph/Network Data

Instructor: Yizhou Sun

yzsun@ccs.neu.edu

November 27, 2014

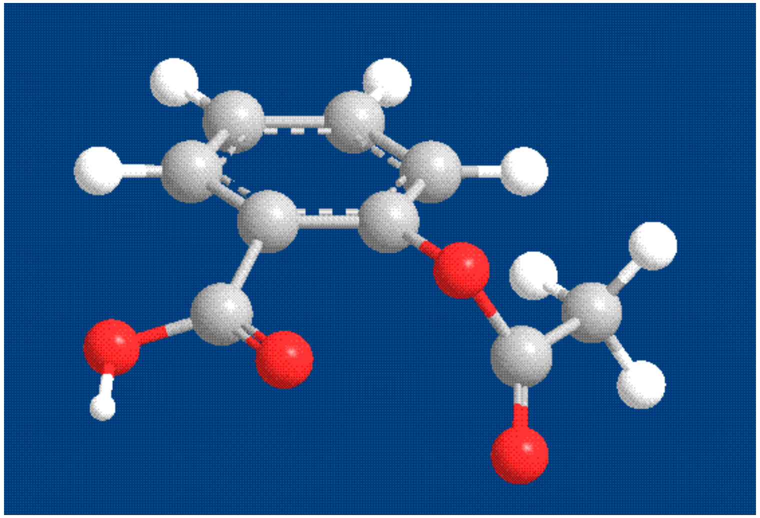
Methods to Learn

	Matrix Data	Set Data	Sequence Data	Time Series	Graph & Network
Classification	Decision Tree; Naïve Bayes; Logistic Regression SVM; kNN		HMM		Label Propagation
Clustering	K-means; hierarchical clustering; DBSCAN; Mixture Models; kernel k-means				SCAN; Spectral Clustering
Frequent Pattern Mining		Apriori; FP-growth	GSP; PrefixSpan		
Prediction	Linear Regression			Autoregression	
Similarity Search				DTW	P-PageRank
Ranking					PageRank

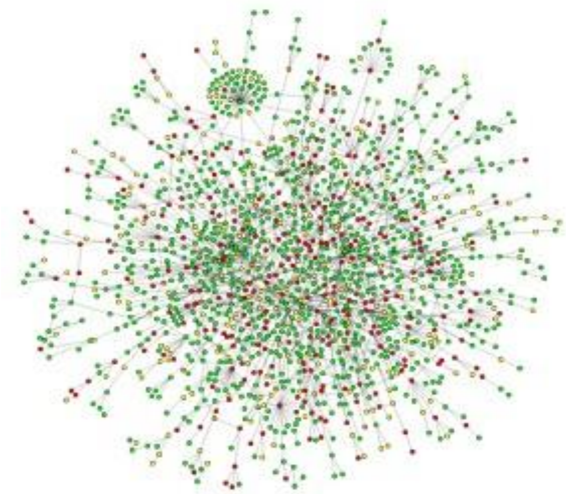
Mining Graph/Network Data

- Graph / Network Data
- Ranking on Graph / Network
- Graph/Network Clustering
- Graph/Network Classification
- Summary

Graph, Graph, Everywhere

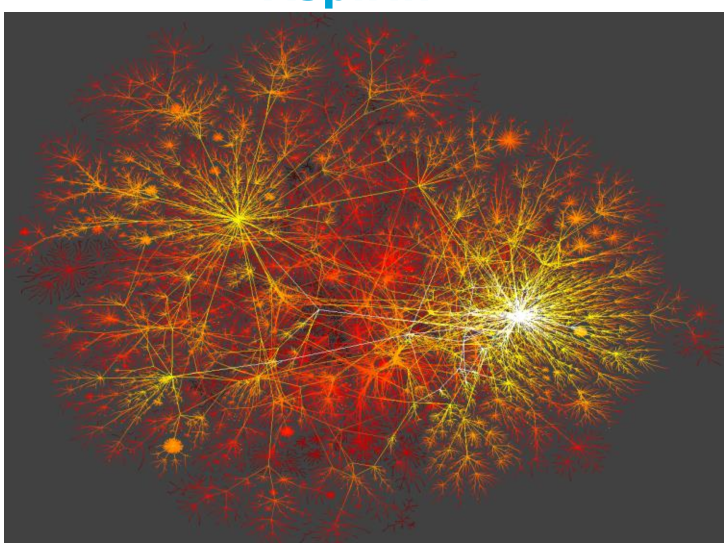


Aspirin

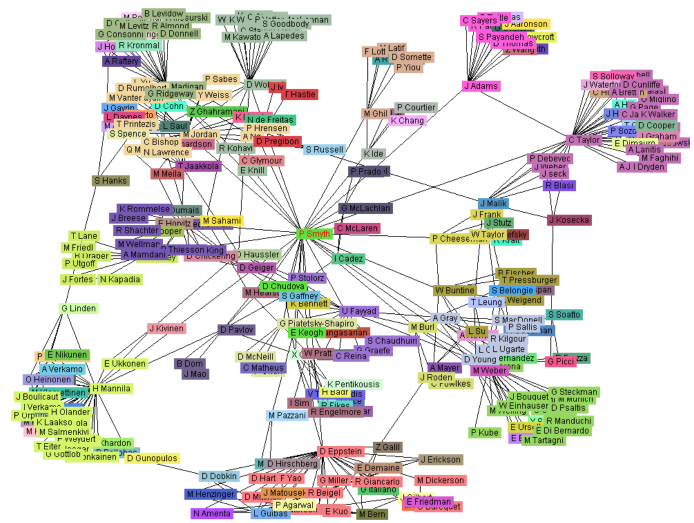


Yeast protein interaction network

from H. Jeong et al Nature 411, 41 (2001)



Internet



Co-author network

Why Graph Mining?

- Graphs are ubiquitous
  - Chemical compounds (Cheminformatics)
  - Protein structures, biological pathways/networks (Bioinformactics)
  - Program control flow, traffic flow, and workflow analysis
  - XML databases, Web, and social network analysis
- Graph is a general model
  - Trees, lattices, sequences, and items are degenerated graphs
- Diversity of graphs
  - Directed vs. undirected, labeled vs. unlabeled (edges & vertices), weighted, with angles & geometry (topological vs. 2-D/3-D)
- Complexity of algorithms: many problems are of high complexity

Representation of a Graph

- $G =< V, E >$
  - $V = \{u_1, \dots, u_n\}$: node set
  - $E \subseteq V \times V$: edge set
- Adjacency matrix
  - $A = \{a_{ij}\}, i, j = 1, \dots, n$
    - $a_{ij} = 1, if \ < u_i, u_j > \in E$
    - $a_{ij} = 0, if \ < u_i, u_j > \notin E$
  - Undirected graph vs. Directed graph
    - $A = A^{\mathrm{T}}\ vs. A \neq A^{\mathrm{T}}$
  - Weighted graph
    - Use *W* instead of *A*, where w_{ij} represents the weight of edge $< u_i, u_j >$

Mining Graph/Network Data

- Graph / Network Data
- Ranking on Graph / Network
- Graph/Network Clustering
- Graph/Network Classification
- Summary

Ranking on Graph / Network

- PageRank

- Personalized PageRank

The History of PageRank

- PageRank was developed by Larry Page (hence the name *Page*-Rank) and Sergey Brin.

- It is first as part of a research project about a new kind of search engine. That project started in 1995 and led to a functional prototype in 1998.

- Shortly after, Page and Brin founded Google.

Ranking web pages

- Web pages are not equally "important"
  - www.cnn.com vs. a personal webpage
- Inlinks as votes
  - The more inlinks, the more important
- Are all inlinks equal?
  - Recursive question!

Simple recursive formulation

- Each link's vote is proportional to the importance of its source page
- If page P with importance x has n outlinks, each link gets x/n votes
- Page P's own importance is the sum of the votes on its inlinks

Matrix formulation

- Matrix **M** has one row and one column for each web page
- Suppose page j has n outlinks
  - If j -> i, then $M_{ij}=1/n$
  - Else $M_{ij}=0$
- **M** is a column stochastic matrix
  - Columns sum to 1
- Suppose **r** is a vector with one entry per web page
  - r_i is the importance score of page i
  - Call it the rank vector
  - $|\mathbf{r}| = 1$

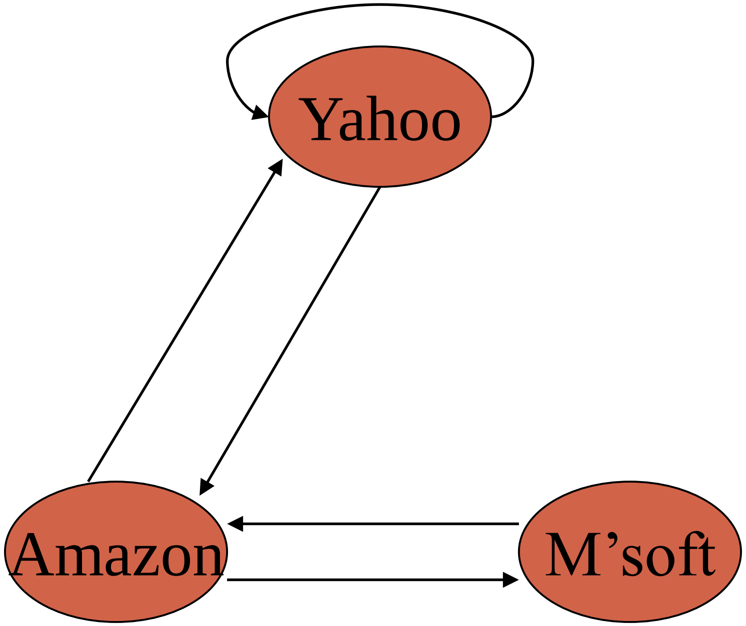
Eigenvector formulation

- The flow equations can be written

$$\mathbf{r} = \mathbf{M}\mathbf{r}$$

- So the rank vector is an eigenvector of the stochastic web matrix
  - In fact, its first or principal eigenvector, with corresponding eigenvalue 1

Example



$$\begin{array}{c|ccc|} & y & a & m \\ y & 1/2 & 1/2 & 0 \\ a & 1/2 & 0 & 1 \\ m & 0 & 1/2 & 0 \end{array}$$

r = Mr

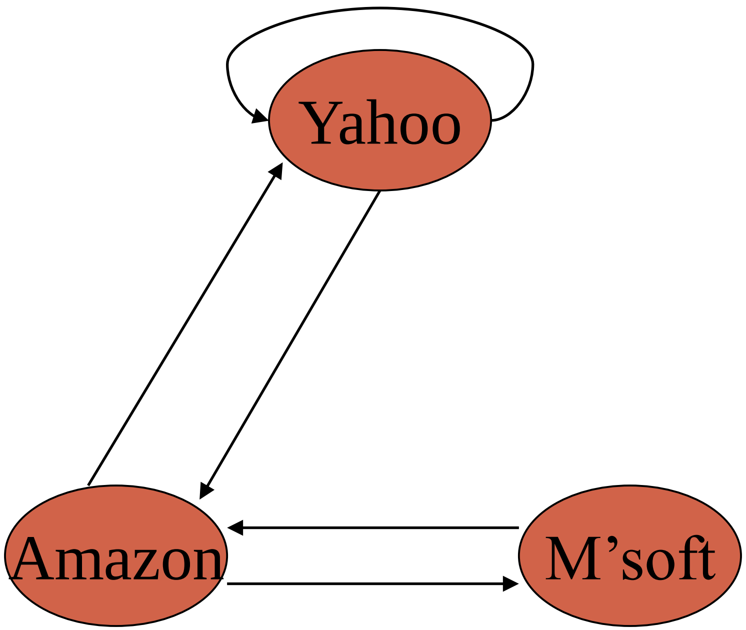
$$y = y/2 + a/2$$
$$a = y/2 + m$$
$$m = a/2$$

$$\begin{bmatrix} y \\ a \\ m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} y \\ a \\ m \end{bmatrix}$$

Power Iteration method

- Simple iterative scheme (aka relaxation)
- Suppose there are N web pages
- Initialize: $\mathbf{r}^0 = [1/N,\ldots.,1/N]^T$
- Iterate: $\mathbf{r}^{k+1} = \mathbf{M}\mathbf{r}^k$
- Stop when $|\mathbf{r}^{k+1} - \mathbf{r}^k|_1 < \varepsilon$
  - $|\mathbf{x}|_1 = \sum_{1 \le i \le N} |x_i|$ is the L_1 norm
  - Can use any other vector norm e.g., Euclidean

Power Iteration Example



$$
\begin{array}{c|ccc|}
 & y & a & m \\
\hline
y & 1/2 & 1/2 & 0 \\
a & 1/2 & 0 & 1 \\
m & 0 & 1/2 & 0 \\
\hline
\end{array}
$$

$$
\begin{array}{c}
y \\ a \\ m
\end{array}
=
\underset{r_0}{\begin{array}{c} 1/3 \\ 1/3 \\ 1/3 \end{array}}
\quad
\underset{r_1}{\begin{array}{c} 1/3 \\ 1/2 \\ 1/6 \end{array}}
\quad
\underset{r_2}{\begin{array}{c} 5/12 \\ 1/3 \\ 1/4 \end{array}}
\quad
\underset{r_3}{\begin{array}{c} 3/8 \\ 11/24 \\ 1/6 \end{array}}
\quad
\underset{\dots}{\dots}
\quad
\underset{r^*}{\begin{array}{c} 2/5 \\ 2/5 \\ 1/5 \end{array}}
$$

Random Walk Interpretation

- Imagine a random web surfer
  - At any time t, surfer is on some page P
  - At time t+1, the surfer follows an outlink from P uniformly at random
  - Ends up on some page Q linked from P
  - Process repeats indefinitely
- Let **p**(t) be a vector whose i^{th} component is the probability that the surfer is at page i at time t
  - **p**(t) is a probability distribution on pages

*The stationary distribution

- Where is the surfer at time t+1?
  - Follows a link uniformly at random
  - $\mathbf{p}(t+1) = \mathbf{M}\mathbf{p}(t)$
- Suppose the random walk reaches a state such that $\mathbf{p}(t+1) = \mathbf{M}\mathbf{p}(t) = \mathbf{p}(t)$
  - Then $\mathbf{p}(t)$ is called a stationary distribution for the random walk
- Our rank vector **r** satisfies $\mathbf{r} = \mathbf{M}\mathbf{r}$
  - So it is a stationary distribution for the random surfer

*Existence and Uniqueness

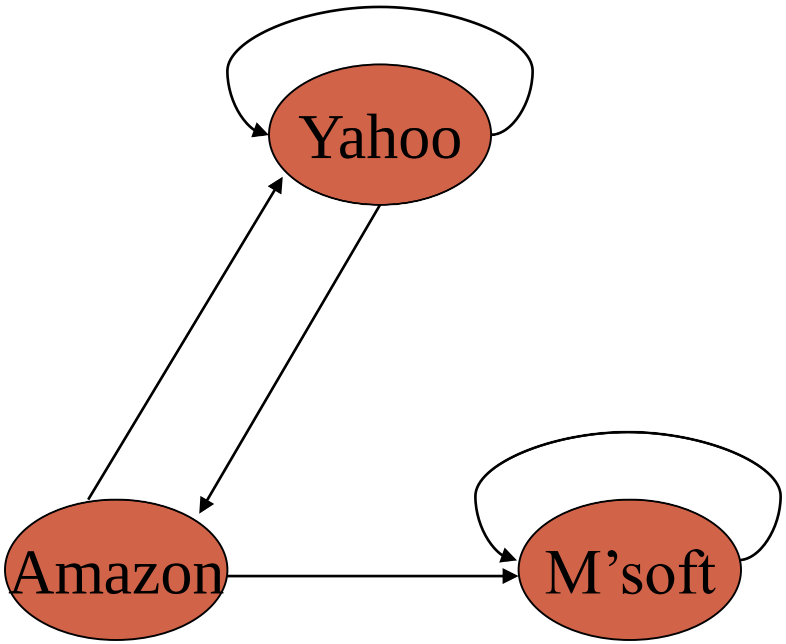
A central result from the theory of random walks (aka Markov processes):

For graphs that satisfy certain conditions, the stationary distribution is unique and eventually will be reached no matter what the initial probability distribution at time t = 0.

Spider traps

- A group of pages is a spider trap if there are no links from within the group to outside the group
  - Random surfer gets trapped
- Spider traps violate the conditions needed for the random walk theorem

Microsoft becomes a spider trap



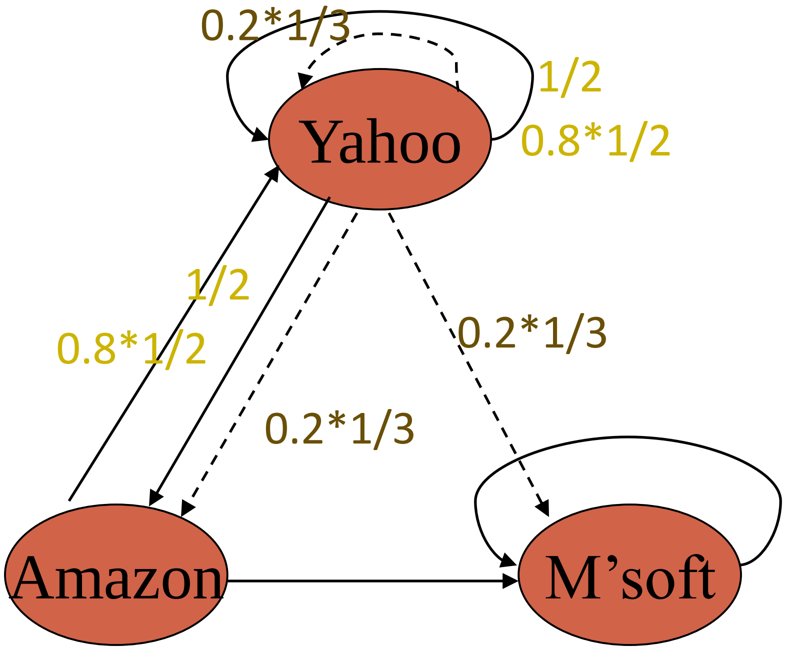
$$
\begin{array}{c|ccc|}
 & y & a & m \\
\hline
y & 1/2 & 1/2 & 0 \\
a & 1/2 & 0 & 0 \\
m & 0 & 1/2 & 1 \\
\hline
\end{array}
$$

$$
\begin{array}{c}
y \\ a \\ m
\end{array}
=
\begin{array}{c}
1/3 \\ 1/3 \\ 1/3
\end{array}
\quad
\begin{array}{c}
1/3 \\ 1/6 \\ 1/2
\end{array}
\quad
\begin{array}{c}
1/4 \\ 1/6 \\ 7/12
\end{array}
\quad
\begin{array}{c}
5/24 \\ 1/8 \\ 2/3
\end{array}
\quad \ldots \quad
\begin{array}{c}
0 \\ 0 \\ 1
\end{array}
$$

Random teleports

- The Google solution for spider traps
- At each time step, the random surfer has two options:
  - With probability β, follow a link at random
  - With probability 1-β, jump to some page uniformly at random
  - Common values for β are in the range 0.8 to 0.9
- Surfer will teleport out of spider trap within a few time steps

Random teleports ($\beta = 0.8$)

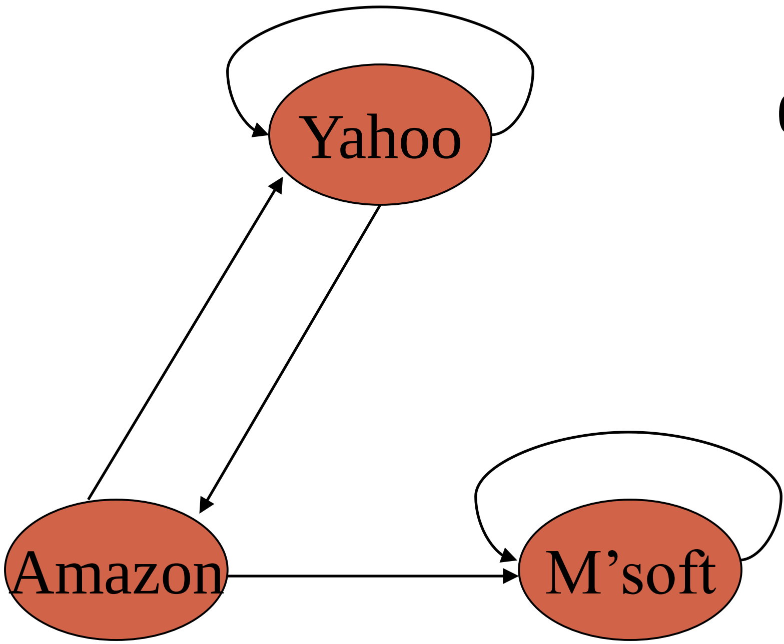


$$
\begin{array}{cc} & \text{y} \\ \begin{array}{c}\text{y}\\ \text{a}\\ \text{m}\end{array} & \boxed{\begin{array}{c}1/2\\ 1/2\\ 0\end{array}} \end{array}
\qquad
0.8* \begin{array}{c} \text{y} \\ \boxed{\begin{array}{c}1/2\\ 1/2\\ 0\end{array}} \end{array}
+ 0.2* \begin{array}{c} \text{y} \\ \boxed{\begin{array}{c}1/3\\ 1/3\\ 1/3\end{array}} \end{array}
$$

$$
0.8 \; \boxed{\begin{array}{ccc} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{array}} \; + 0.2 \; \boxed{\begin{array}{ccc} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{array}}
$$

$$
\begin{array}{cc} \begin{array}{c}\text{y}\\ \text{a}\\ \text{m}\end{array} & \boxed{\begin{array}{ccc} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{array}} \end{array}
$$

Random teleports ($\beta = 0.8$)



$$0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

$$\begin{array}{c} y \\ a \\ m \end{array} \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}$$

$$\begin{array}{c} y \\ a \\ m \end{array} = \begin{bmatrix} 0.333 \\ 0.333 \\ 0.333 \end{bmatrix} \begin{bmatrix} 0.333 \\ 0.200 \\ 0.467 \end{bmatrix} \begin{bmatrix} 0.280 \\ 0.200 \\ 0.520 \end{bmatrix} \begin{bmatrix} 0.259 \\ 0.179 \\ 0.563 \end{bmatrix} \ldots \begin{bmatrix} 7/33 \\ 5/33 \\ 21/33 \end{bmatrix}$$

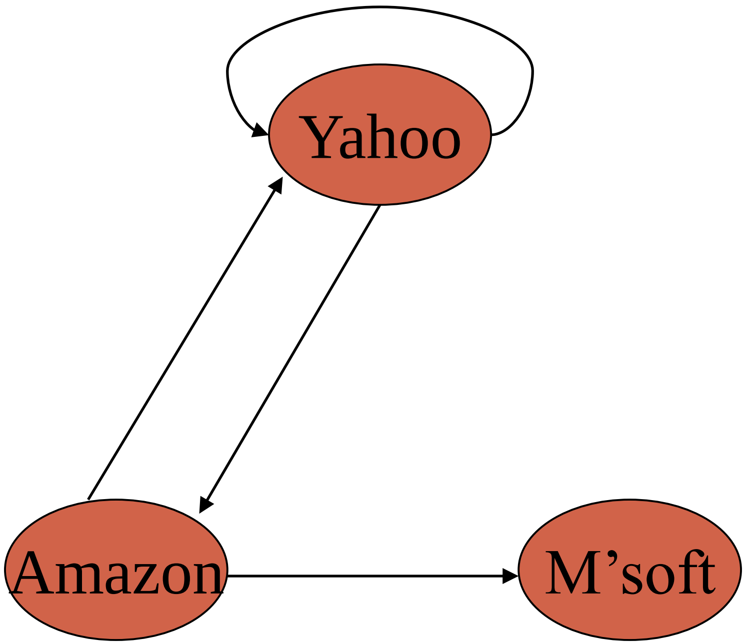
PageRank

- Construct the N-by-N matrix **A** as follows
  - $M^*_{ij} = \beta M_{ij} + (1-\beta)/N$
- Verify that $\mathbf{M}^*$ is a stochastic matrix
- The page rank vector **r** is the principal eigenvector of this matrix
  - satisfying $\mathbf{r} = \mathbf{M}^*\mathbf{r}$
- Equivalently, **r** is the stationary distribution of the random walk with teleports

Dead ends

- Pages with no outlinks are “dead ends” for the random surfer
  - Nowhere to go on next step

Microsoft becomes a dead end



$$0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

$$\begin{array}{c} y \\ a \\ m \end{array} \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 1/15 \end{bmatrix}$$

Non-stochastic!

$$\begin{array}{c} y \\ a \\ m \end{array} = \begin{array}{c} 1/3 \\ 1/3 \\ 1/3 \end{array} \quad \begin{array}{c} 1/3 \\ 0.2 \\ 0.2 \end{array} \quad \ldots \quad \begin{array}{c} 0 \\ 0 \\ 0 \end{array}$$

Dealing with dead-ends

- Teleport
  - Follow random teleport links with probability 1.0 from dead-ends
  - Adjust matrix accordingly
- Prune and propagate
  - Preprocess the graph to eliminate dead-ends
  - Might require multiple passes
  - Compute page rank on reduced graph
  - Approximate values for deadends by propagating values from reduced graph

Computing PageRank

- Key step is matrix-vector multiplication
  - $\mathbf{r}^{\text{new}} = \mathbf{M}^{*}\mathbf{r}^{\text{old}}$
- Easy if we have enough main memory to hold $\mathbf{M}^{*}$, $\mathbf{r}^{\text{old}}$, $\mathbf{r}^{\text{new}}$
- Say N = 1 billion pages
  - We need 4 bytes for each entry (say)
  - 2 billion entries for vectors, approx 8GB
  - Matrix $\mathbf{M}^{*}$ has N^2 entries
    - 10^{18} is a large number!

Rearranging the equation

$\mathbf{r} = \mathbf{M}^*\mathbf{r}$, where

$M^*_{ij} = \beta M_{ij} + (1-\beta)/N$

$r_i = \sum_{1\leq j\leq N} M^*_{ij}\, r_j$

$r_i = \sum_{1\leq j\leq N} [\beta M_{ij} + (1-\beta)/N]\, r_j$

$\quad = \beta \sum_{1\leq j\leq N} M_{ij}\, r_j + (1-\beta)/N \sum_{1\leq j\leq N} r_j$

$\quad = \beta \sum_{1\leq j\leq N} M_{ij}\, r_j + (1-\beta)/N$, since $|\mathbf{r}| = 1$

$\mathbf{r} = \beta\mathbf{M}\mathbf{r} + [(1-\beta)/N]_N$

where $[x]_N$ is an N-vector with all entries x

Sparse matrix formulation

- We can rearrange the page rank equation:
  - $\mathbf{r} = \beta\mathbf{M}\mathbf{r} + [(1-\beta)/N]_N$
  - $[(1-\beta)/N]_N$ is an N-vector with all entries $(1-\beta)/N$
- **M** is a sparse matrix!
  - 10 links per node, approx 10N entries
- So in each iteration, we need to:
  - Compute $\mathbf{r}^{new} = \beta\mathbf{M}\mathbf{r}^{old}$
  - Add a constant value $(1-\beta)/N$ to each entry in $\mathbf{r}^{new}$

Sparse matrix encoding

- Encode sparse matrix using only nonzero entries
  - Space proportional roughly to number of links
  - say 10N, or 4*10*1 billion = 40GB
  - still won't fit in memory, but will fit on disk

source node	degree	destination nodes
0	3	1, 5, 7
1	5	17, 64, 113, 117, 245
2	2	13, 23

Personalized PageRank

- Query-dependent Ranking
  - For a query webpage q, which webpages are most important to q?
  - The relative important webpages to different queries would be different

Calculation of P-PageRank

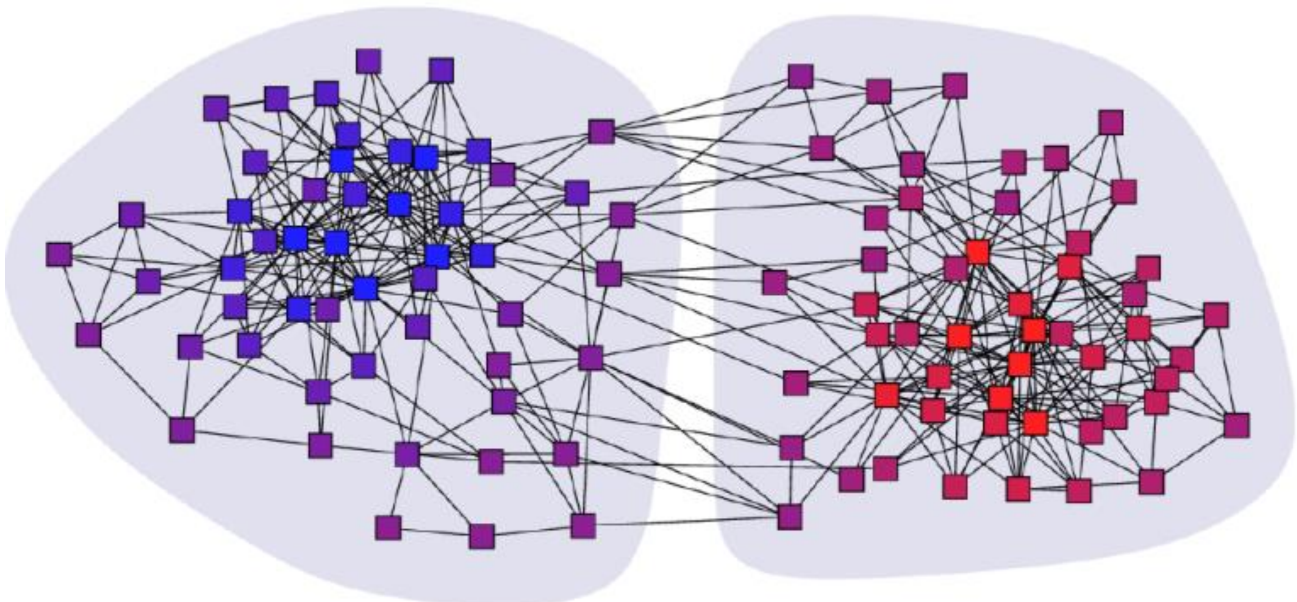
- Recall PageRank calculation:
  - $\mathbf{r} = \beta\mathbf{M}\mathbf{r} + [(1-\beta)/N]_N$ or
  - $\mathbf{r} = \beta\mathbf{M}\mathbf{r} + (1-\beta)\, q_0$, where $q_0 = \begin{pmatrix} 1/N \\ 1/N \\ \dots \\ 1/N \end{pmatrix}$
- For P-PageRank
  - Replace q_0 with $q_0 = \begin{pmatrix} 0 \\ 0 \\ \dots \\ 1 \\ \dots \\ 0 \end{pmatrix}$ qth webpage

Mining Graph/Network Data

- Graph / Network Data
- Ranking on Graph / Network
- Graph/Network Clustering
- Graph/Network Classification
- Summary

Clustering Graphs and Network Data

- Applications
  - Bi-partite graphs, e.g., customers and products, authors and conferences
  - Web search engines, e.g., click through graphs and Web graphs
  - Social networks, friendship/coauthor graphs



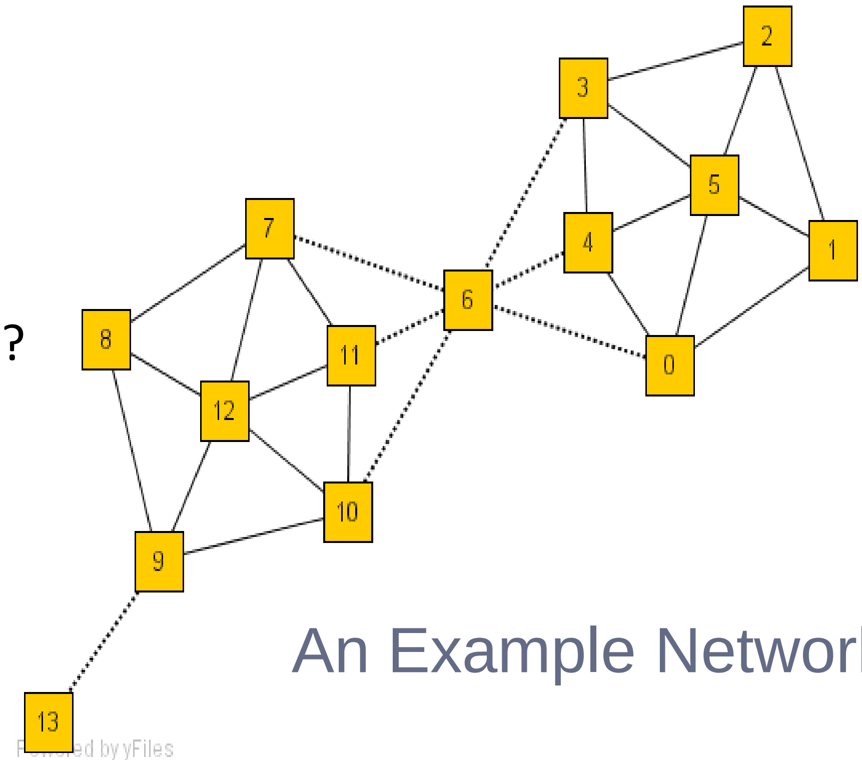
Clustering books about politics [Newman, 2006]

Algorithms

- Graph clustering methods
  - Density-based clustering: SCAN (Xu et al., KDD'2007)
  - Spectral clustering
  - Modularity-based approach
  - Probabilistic approach
  - Nonnegative matrix factorization
  - …

SCAN: Density-Based Clustering of Networks

- How many clusters?
- What size should they be?
- What is the best partitioning?
- Should some points be segregated?

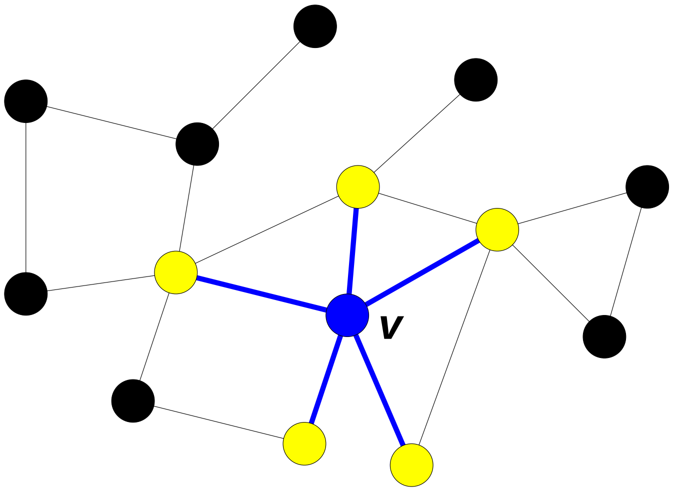


An Example Network

- Application: Given simply information of who associates with whom, could one identify clusters of individuals with common interests or special relationships (families, cliques, terrorist cells)?

A Social Network Model

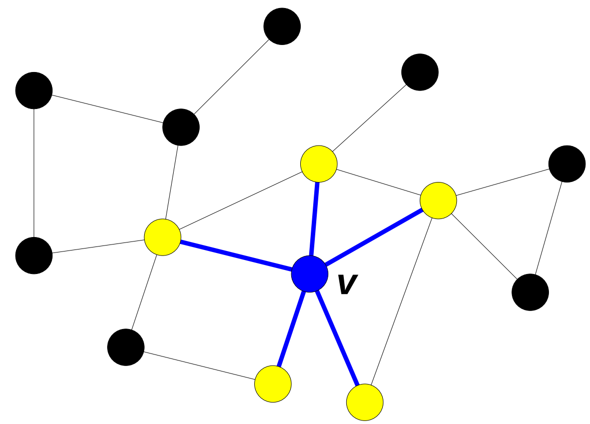
- Cliques, hubs and outliers
  - Individuals in a tight social group, or clique, know many of the same people, regardless of the size of the group
  - Individuals who are hubs know many people in different groups but belong to no single group. Politicians, for example bridge multiple groups
  - Individuals who are outliers reside at the margins of society. Hermits, for example, know few people and belong to no group
- The Neighborhood of a Vertex
  - Define $\Gamma(v)$ as the immediate neighborhood of a vertex (i.e. the set of people that an individual knows)



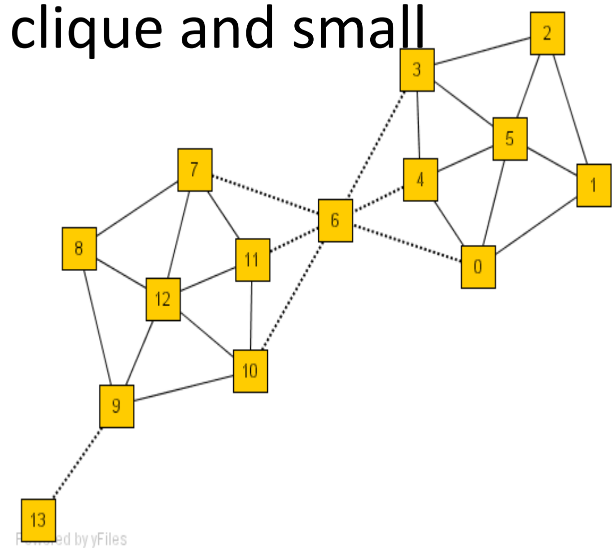
Structure Similarity

- The desired features tend to be captured by a measure we call Structural Similarity

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



- Structural similarity is large for members of a clique and small for hubs and outliers



Structural Connectivity [1]

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

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

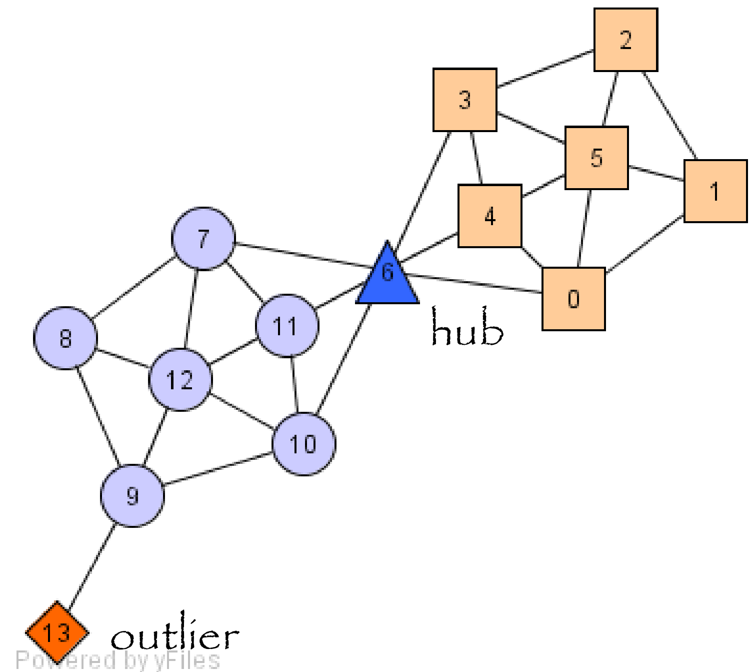
- Structure reachable: transitive closure of direct structure reachability
- Structure connected:

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

[1] M. Ester, H. P. Kriegel, J. Sander, & X. Xu (KDD'96) “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases

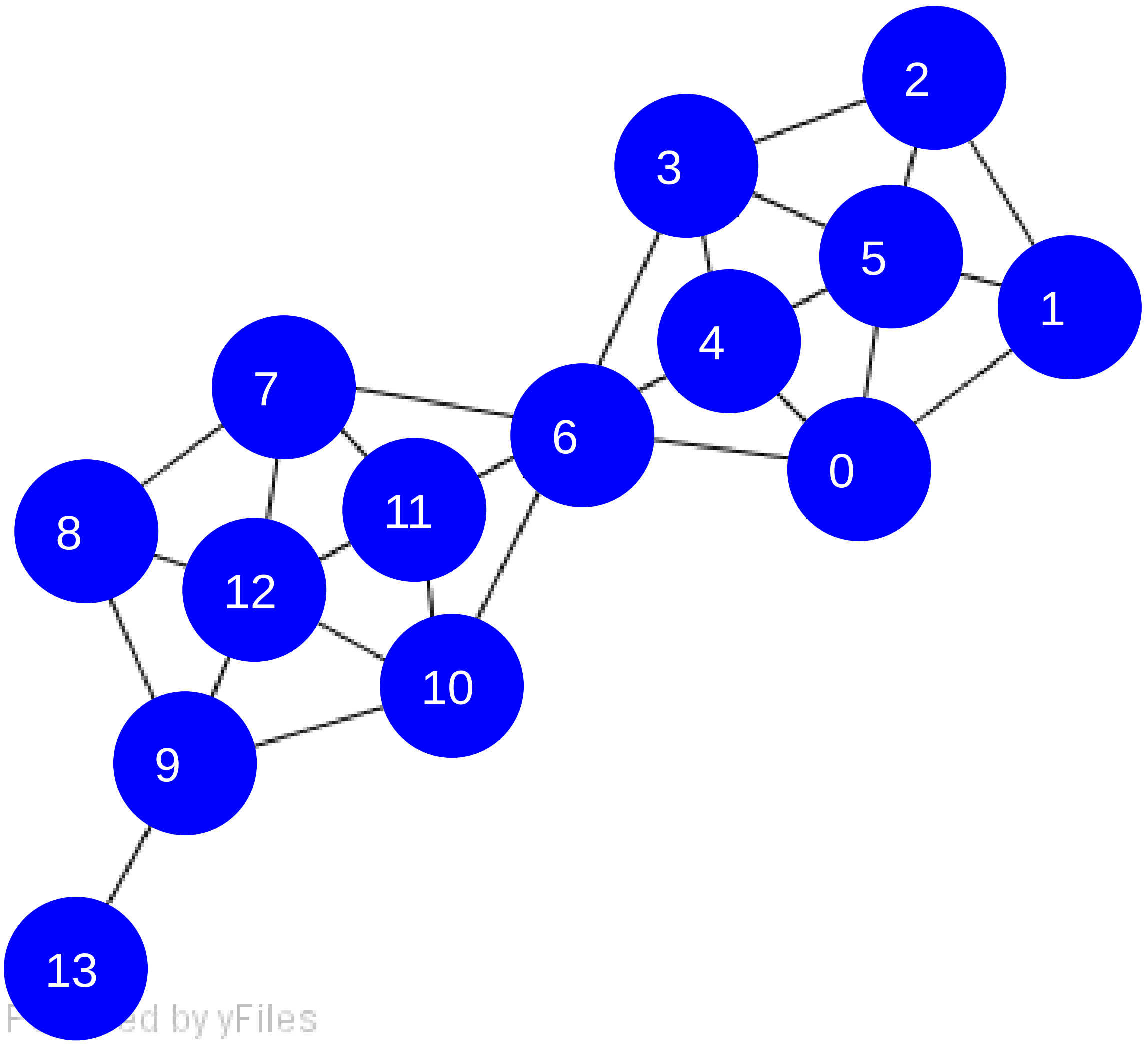
Structure-Connected Clusters

- 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:
  - Not belong to any cluster
  - Bridge to many clusters
- Outliers:
  - Not belong to any cluster
  - Connect to less clusters



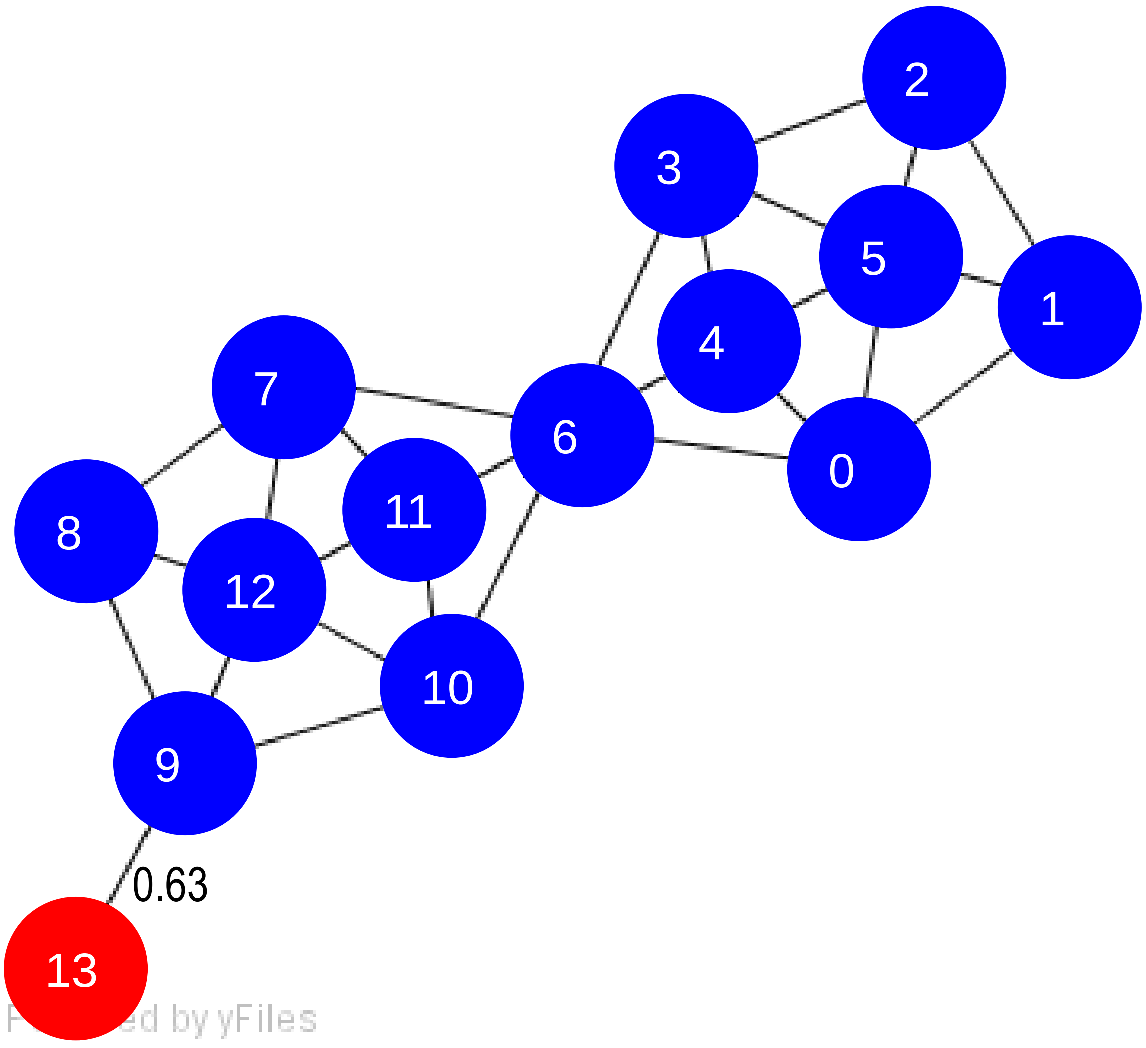
Algorithm

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



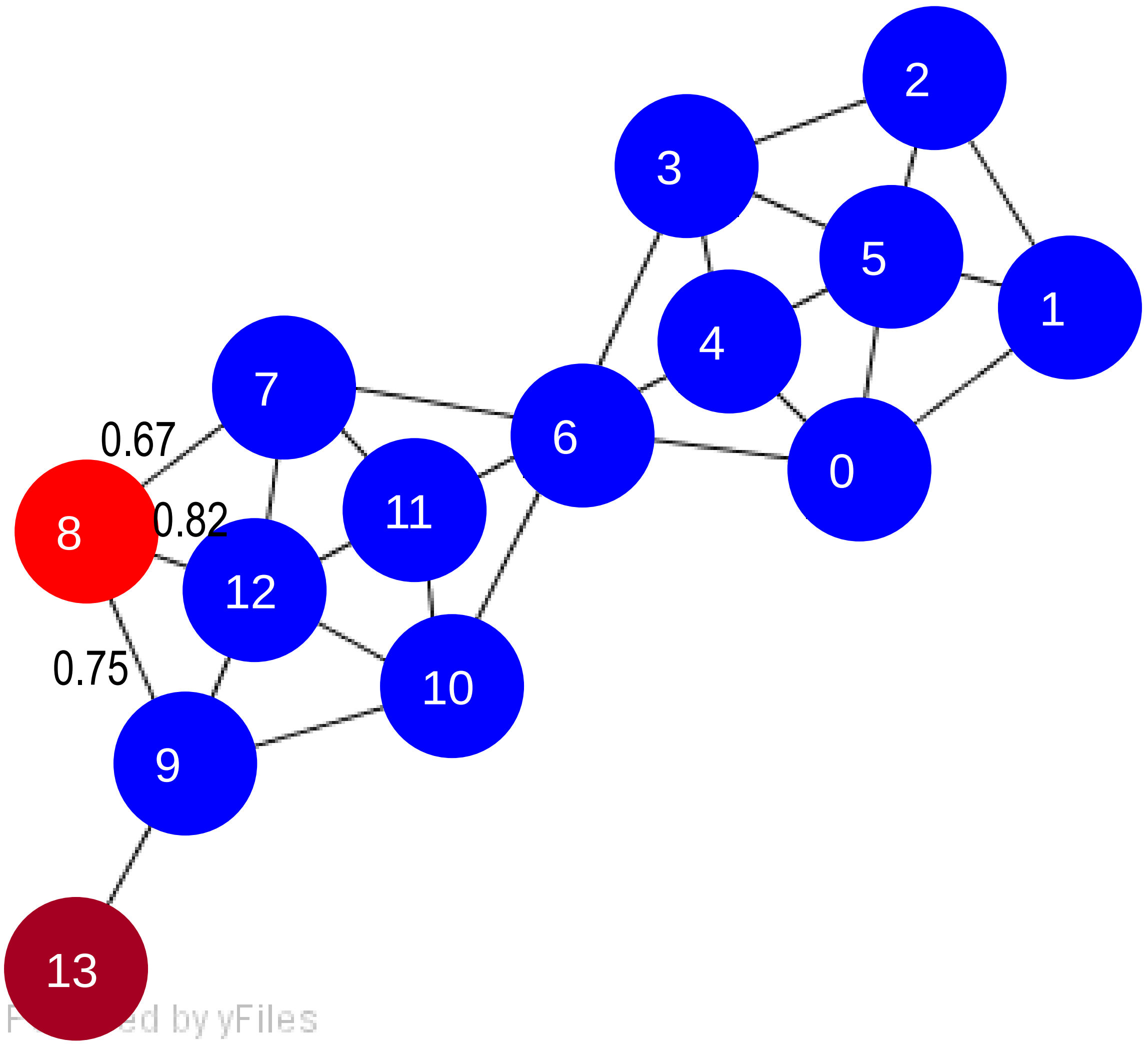
Algorithm

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



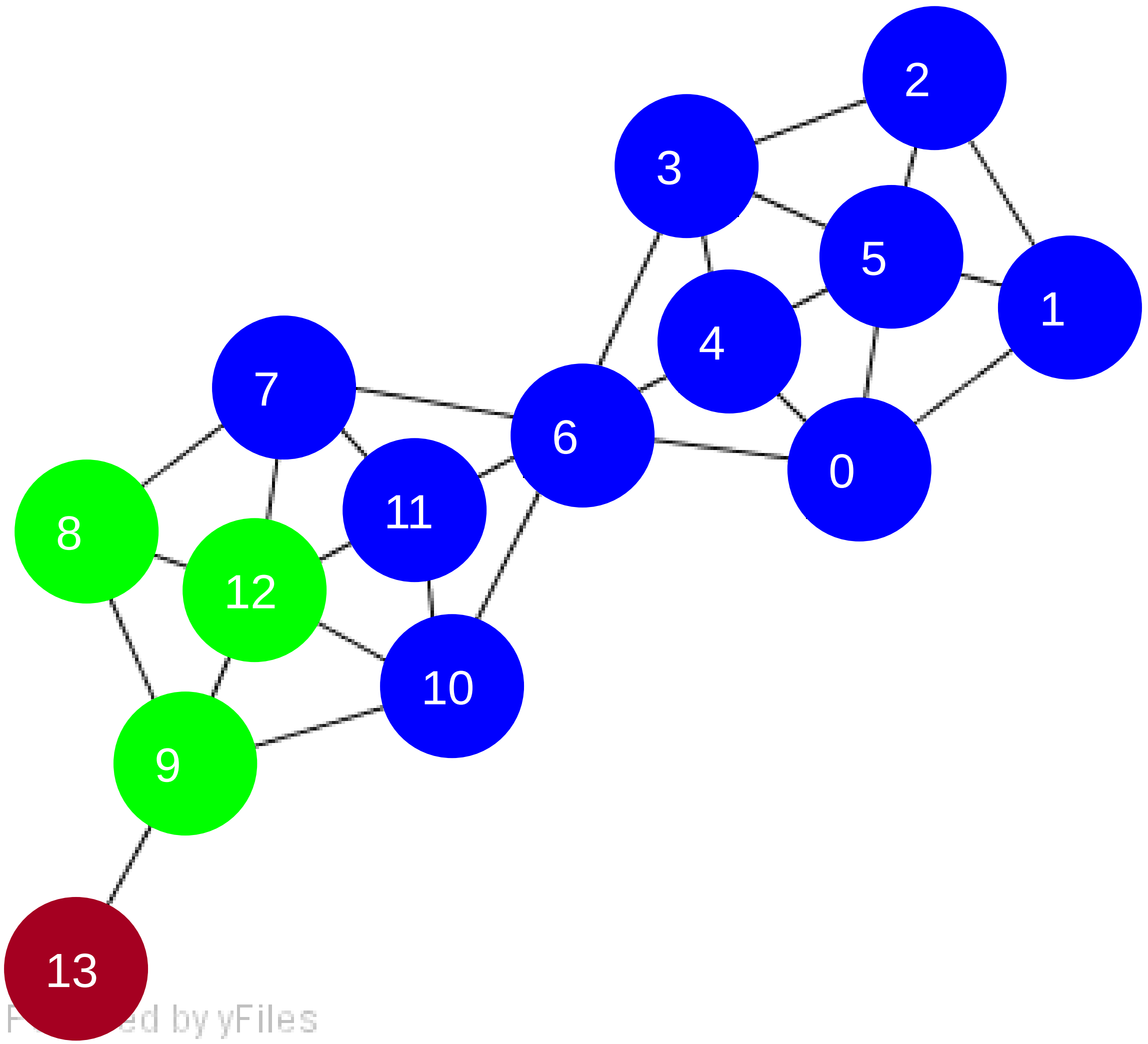
Algorithm

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



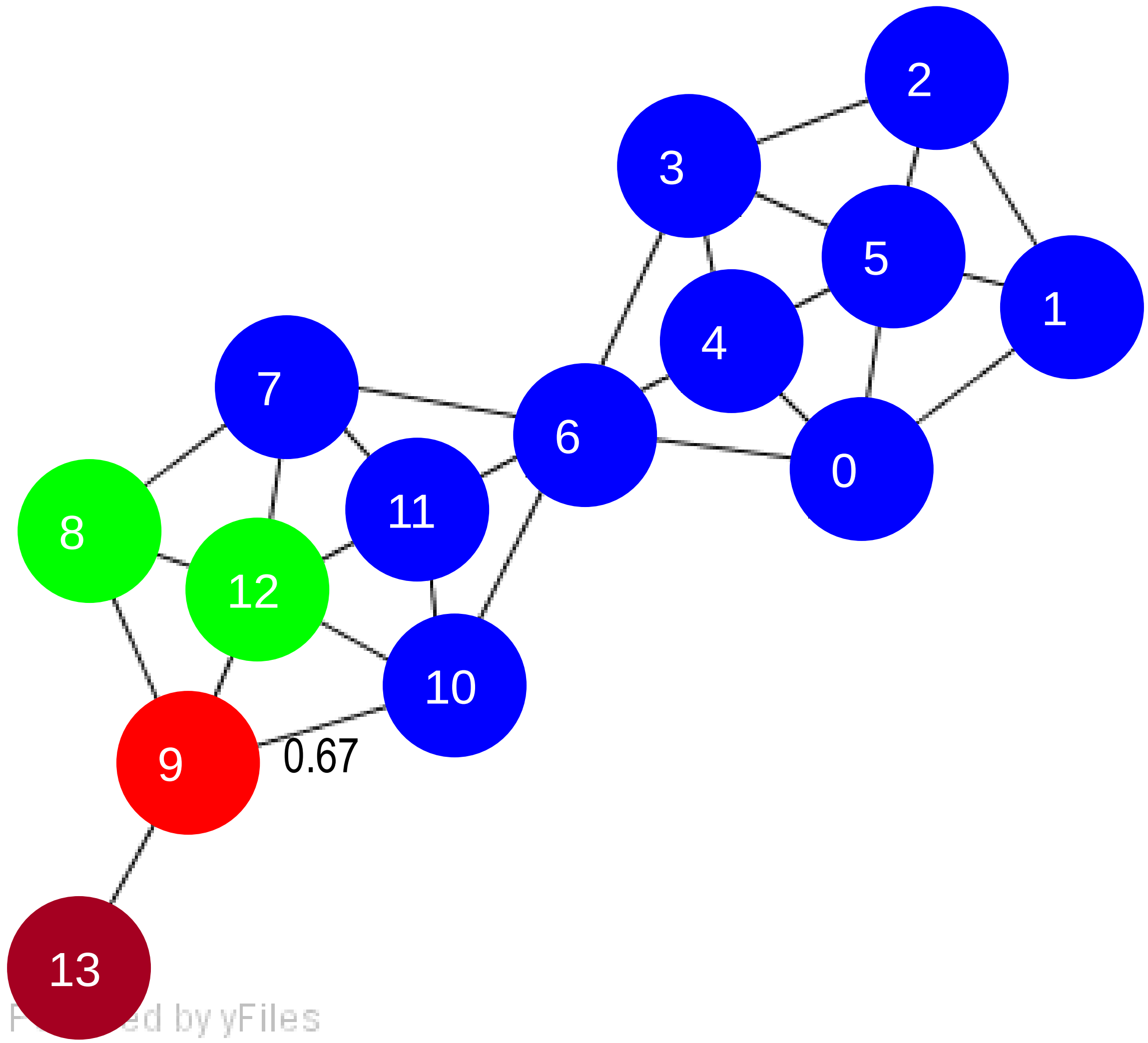
Algorithm

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



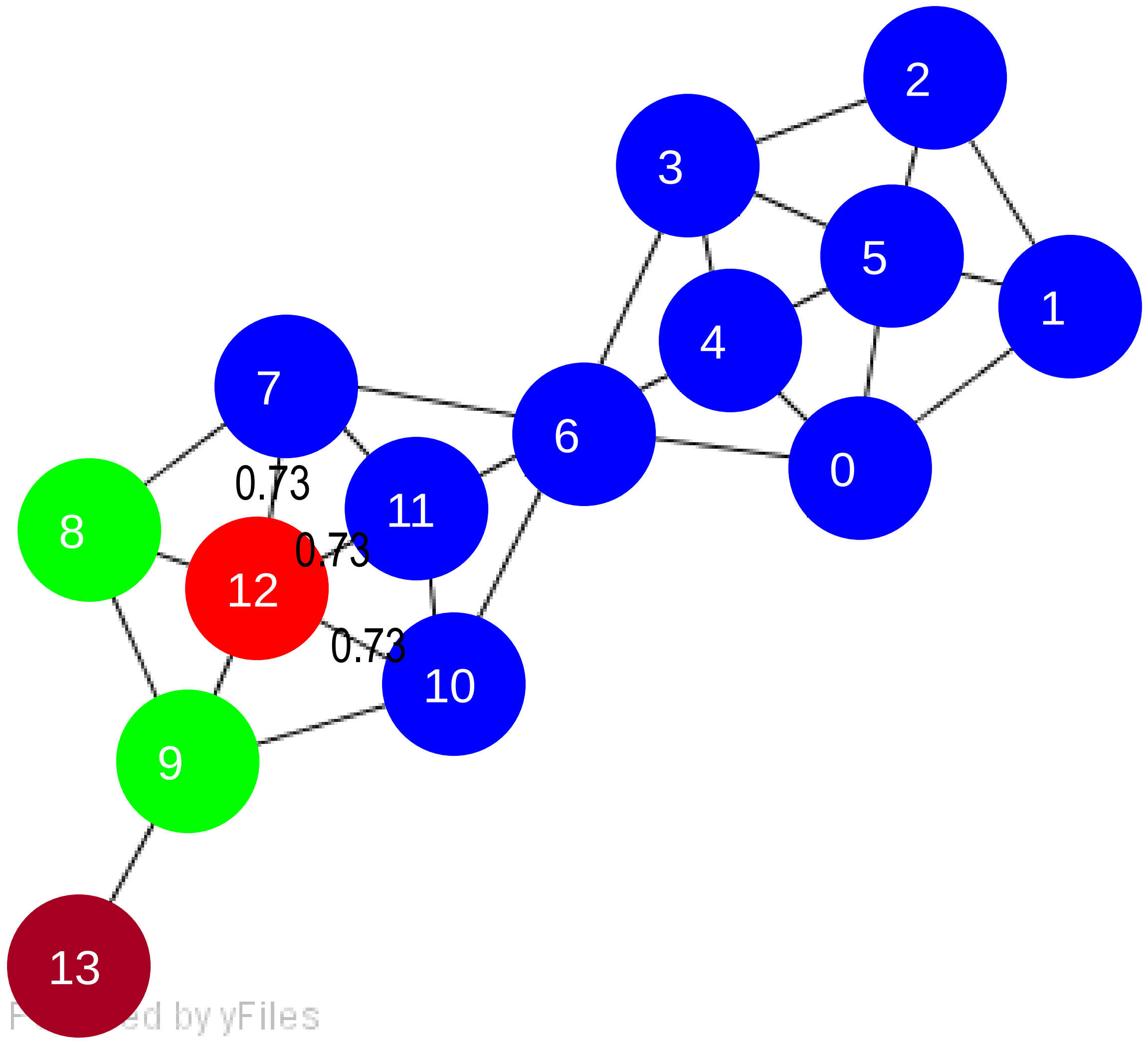
Algorithm

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



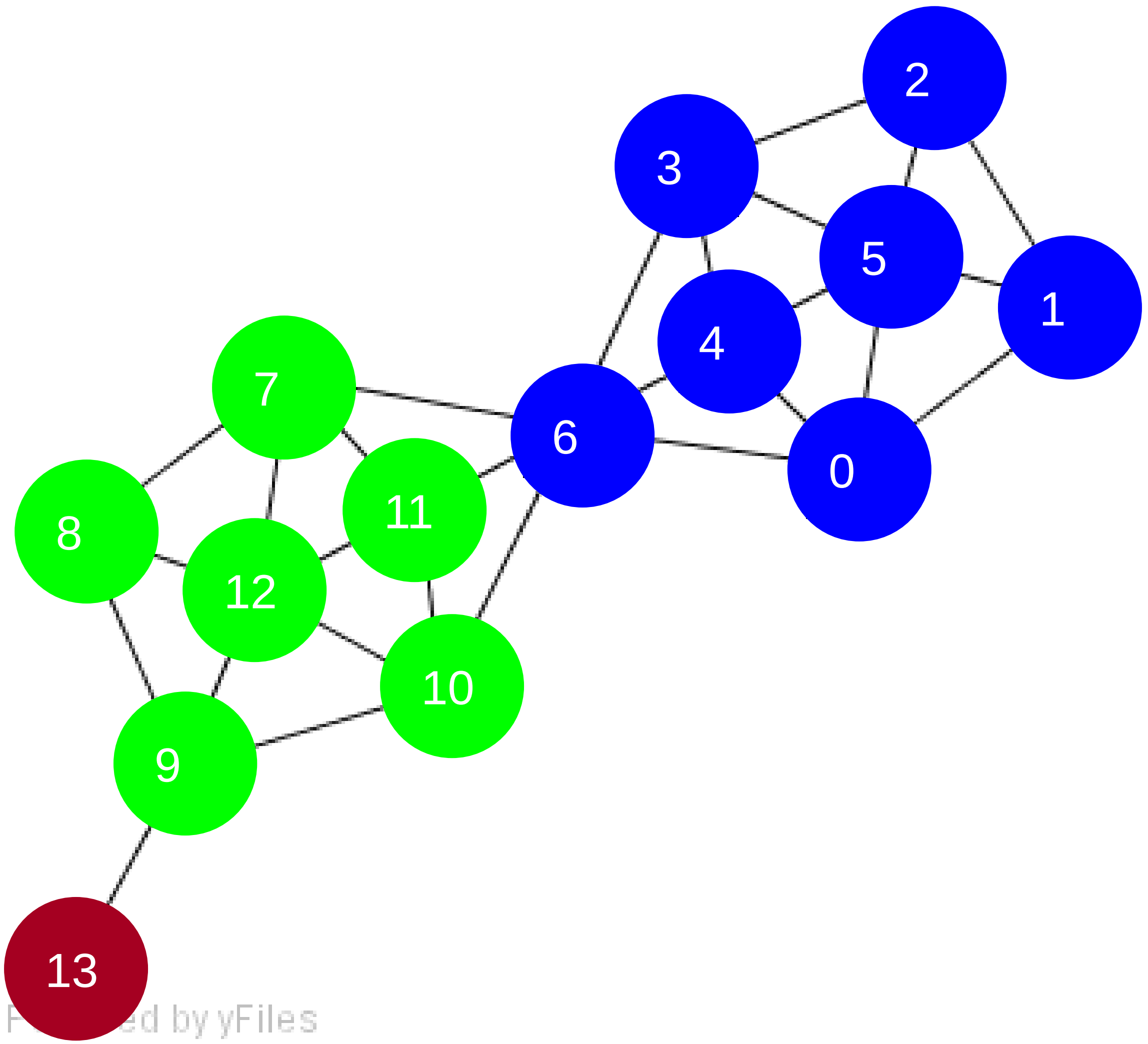
Algorithm

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



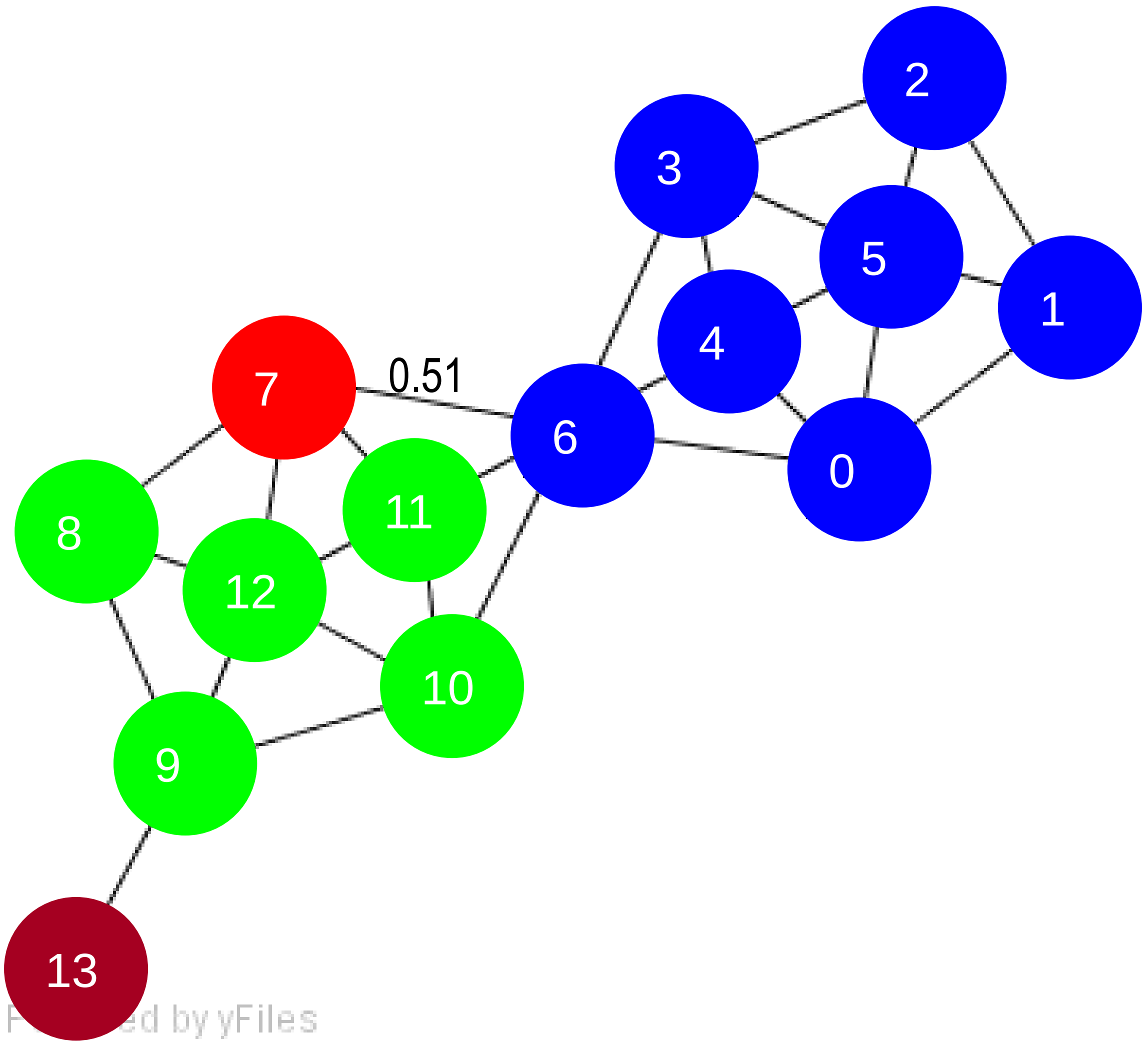
Algorithm

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



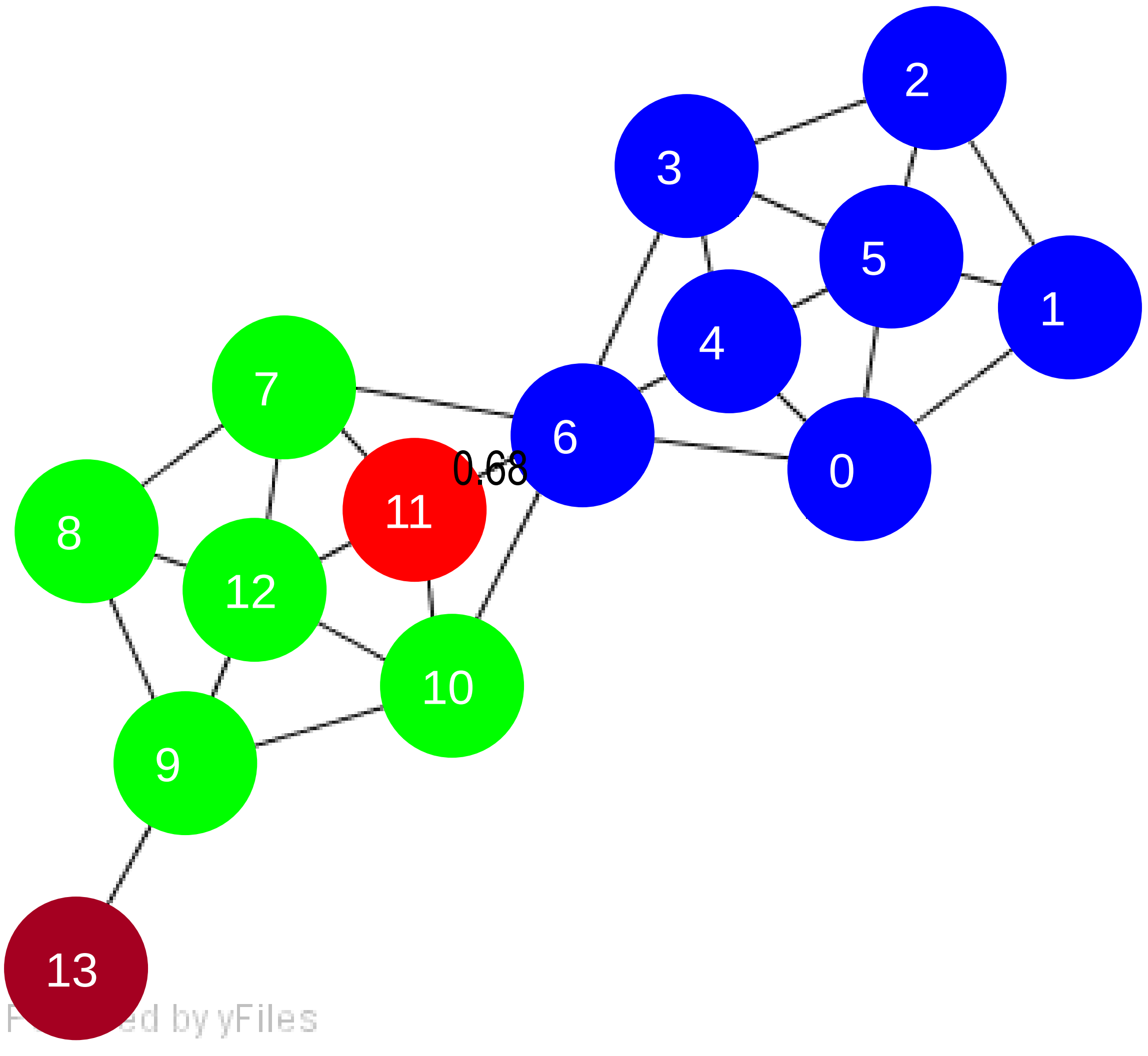
Algorithm

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



Algorithm

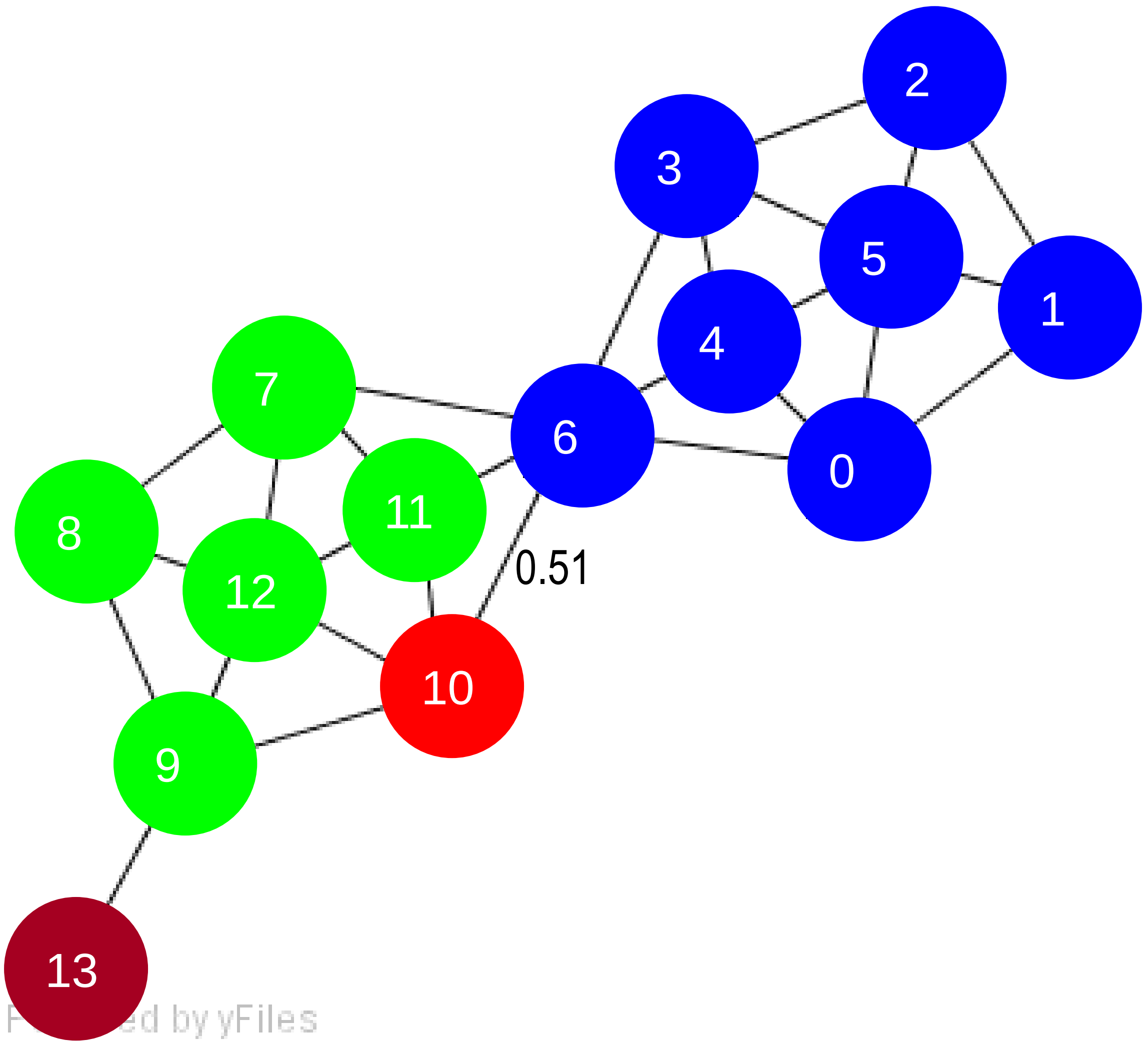
$\mu = 2$
$\varepsilon = 0.7$



Algorithm

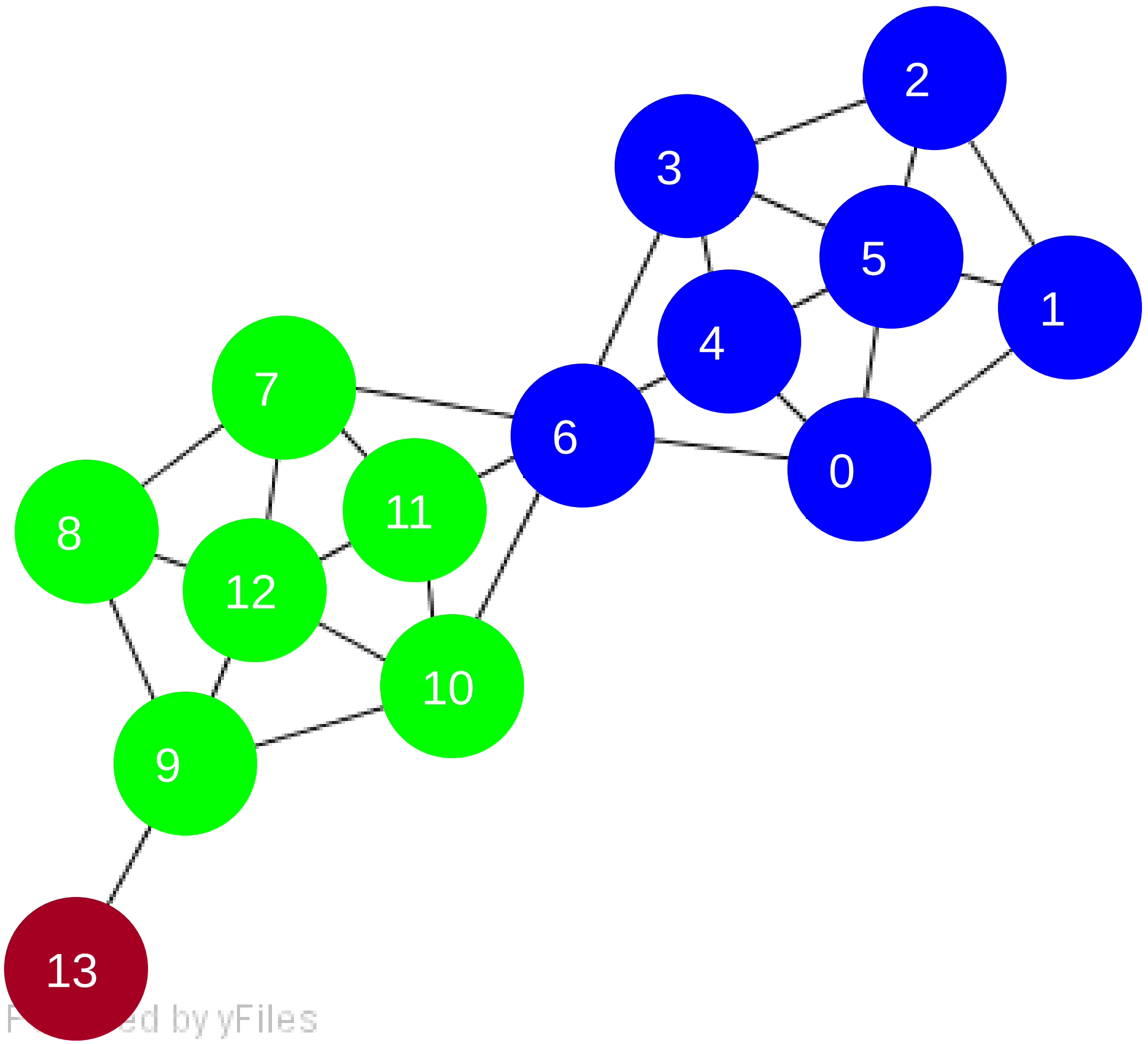
$\mu = 2$

$\varepsilon = 0.7$



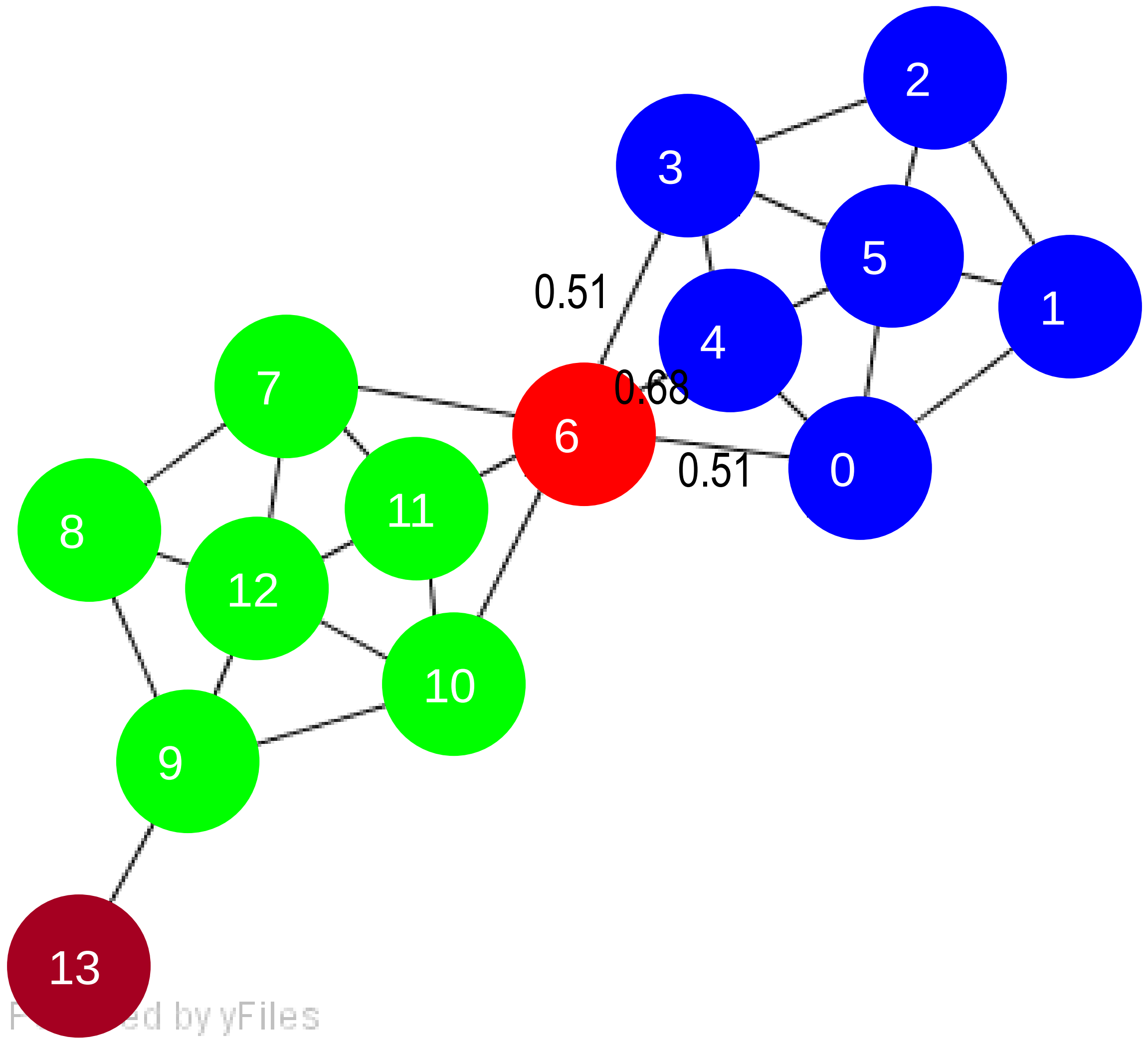
Algorithm

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



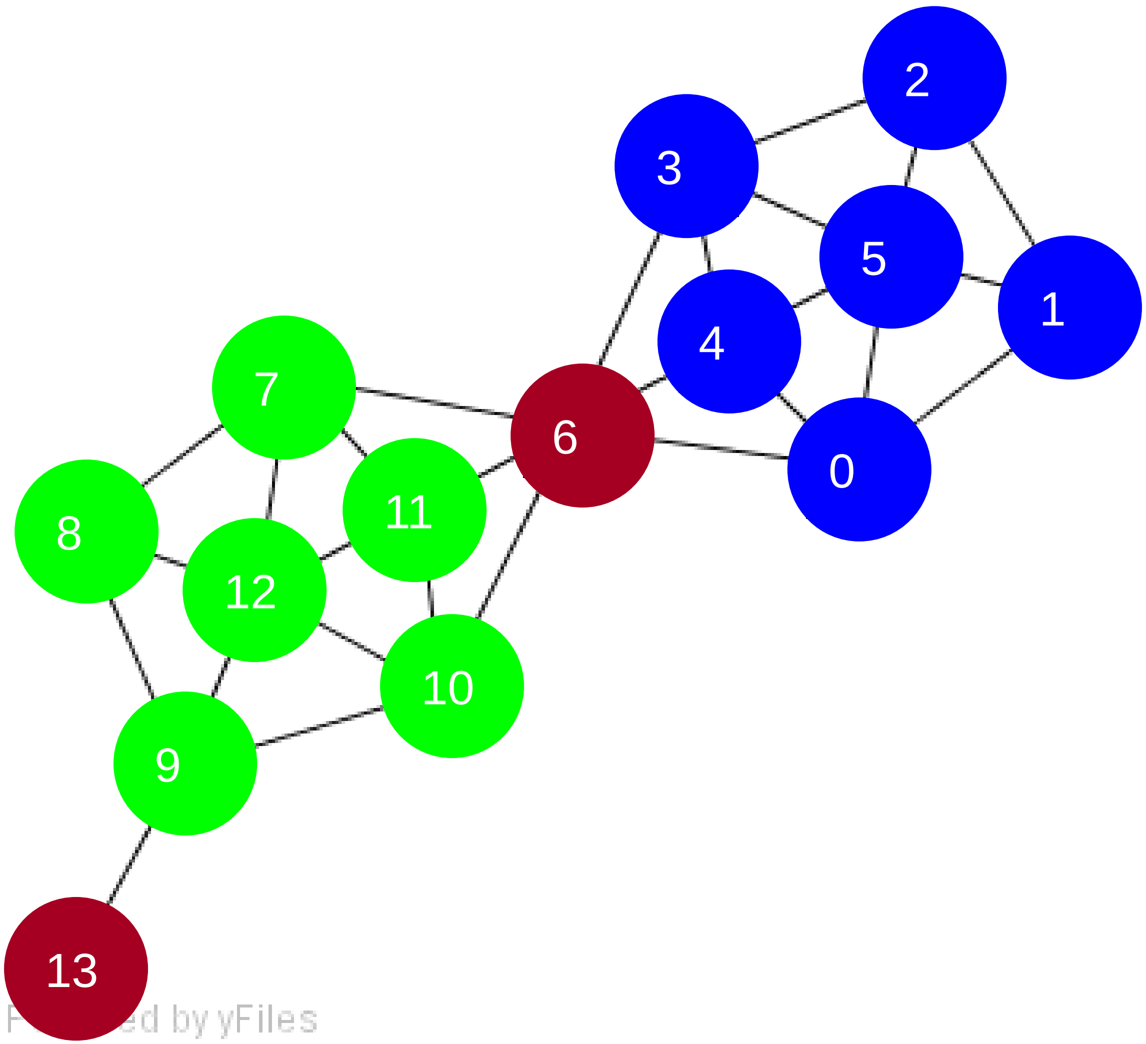
Algorithm

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



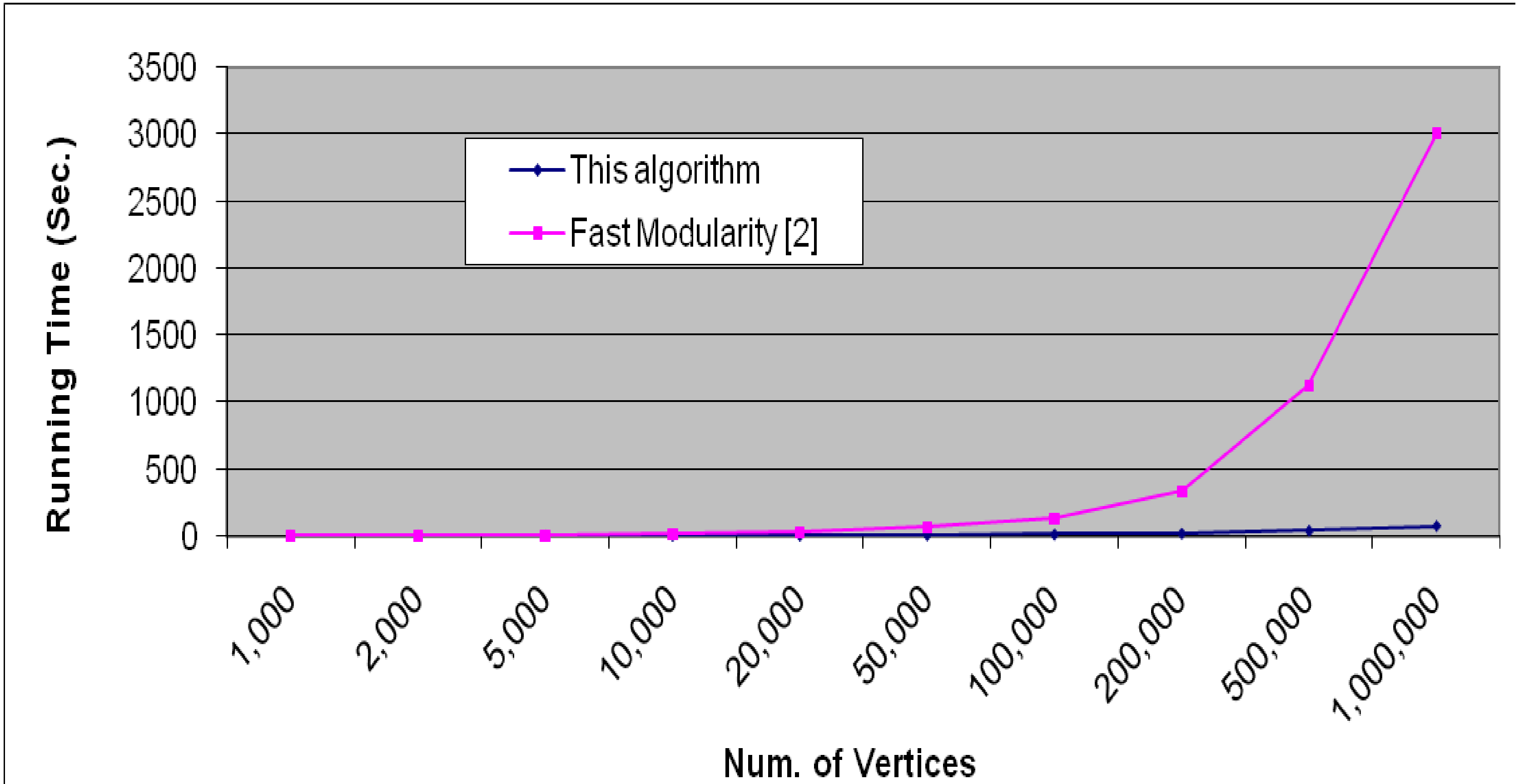
Algorithm

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



Running Time

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



[2] A. Clauset, M. E. J. Newman, & C. Moore, *Phys. Rev. E* **70**, 066111 (2004).

Spectral Clustering

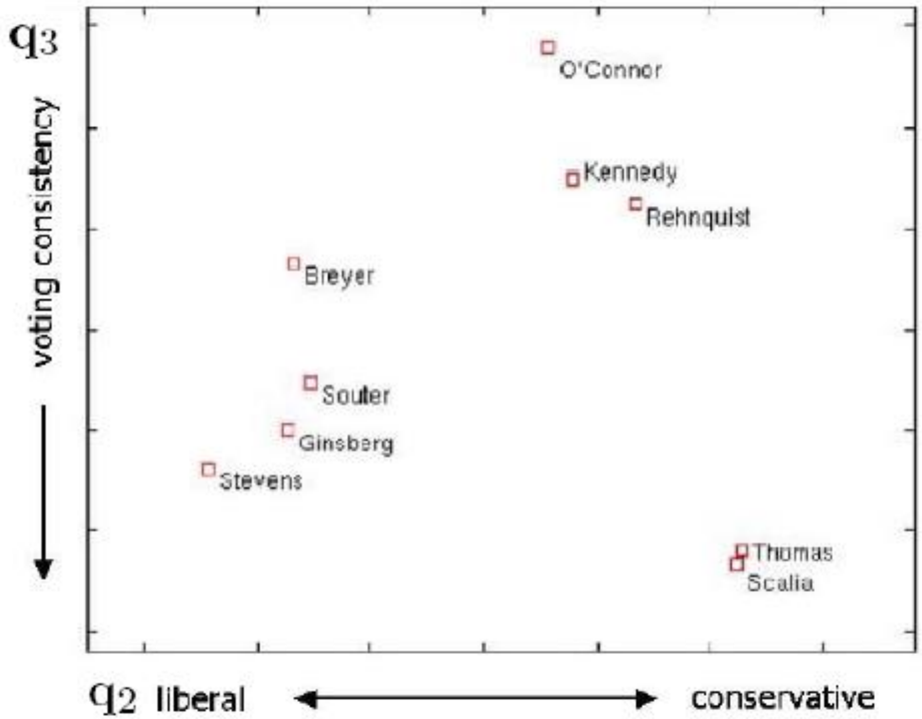
- Reference: ICDM'09 Tutorial by Chris Ding
- Example:
  - Clustering supreme court justices according to

Number of times (%) two Justices voted in agreement

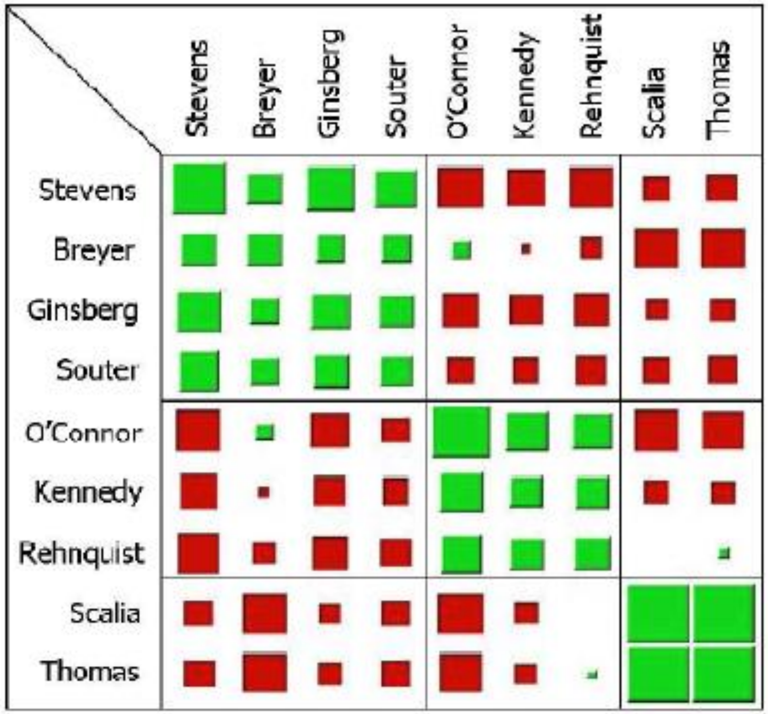
	Ste	Bre	Gin	Sou	O'Co	Ken	Reh	Sca	Tho
Stevens	–	62	66	63	33	36	25	14	15
Breyer	62	–	72	71	55	47	43	25	24
Ginsberg	66	72	–	78	47	49	43	28	26
Souter	63	71	78	–	55	50	44	31	29
O'Connor	33	55	47	55	–	67	71	54	54
Kennedy	36	47	49	50	67	–	77	58	59
Rehnquist	25	43	43	44	71	77	–	66	68
Scalia	14	25	28	31	54	58	66	–	79
Thomas	15	24	26	29	54	59	68	79	–

Table 1: From the voting record of Justices 1995 Term – 2004 Term, the number of times two justices voted in agreement (in percentage). (Data source: from July 2, 2005 *New York Times*. Originally from *Legal Affairs*; *Harvard Law Review*)

Example: Continue



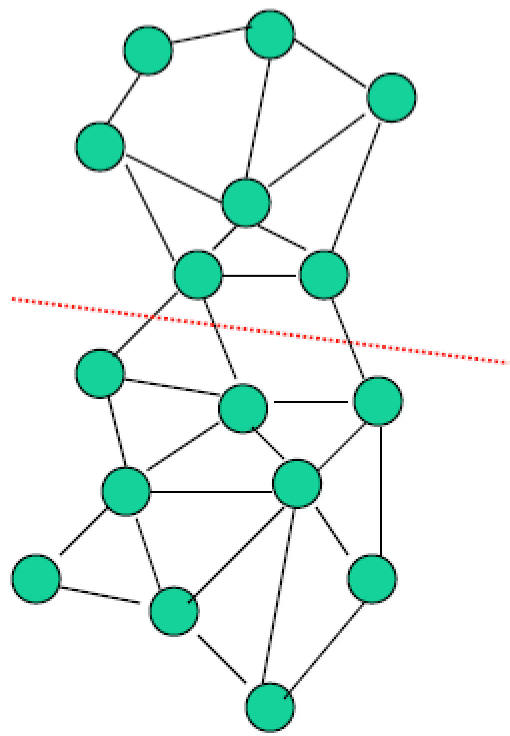
$$C = \mathbf{q}_2\mathbf{q}_2^T + \mathbf{q}_3\mathbf{q}_3^T$$



- Three groups in the Supreme Court:
  - Left leaning group, center-right group, right leaning group.

Spectral Graph Partition

- Min-Cut
  - Minimize the # of cut of edges



Objective Function

2-way Spectral Graph Partitioning

Partition membership indicator: $q_i = \begin{cases} 1 & \textbf{if } \ i \in A \\ -1 & \textbf{if } \ i \in B \end{cases}$

$$J = CutSize = \frac{1}{4}\sum_{i,j} w_{ij}[q_i - q_j]^2$$

$$= \frac{1}{4}\sum_{i,j} w_{ij}[q_i^2 + q_j^2 - 2q_i q_j] = \frac{1}{2}\sum_{i,j} q_i[d_i\delta_{ij} - w_{ij}]q_j$$

$$= \frac{1}{2} q^T (D - W) q$$

Relax indicators q_i from discrete values to continuous values, the solution for min $J(q)$ is given by the eigenvectors of

$$(D - W)q = \lambda q$$

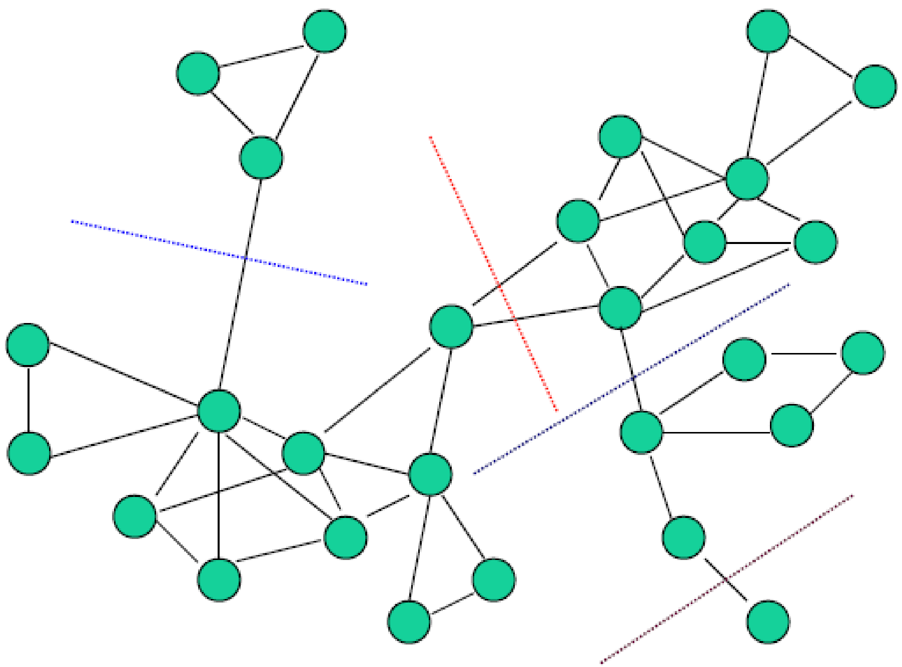
(Fiedler, 1973, 1975)

(Pothen, Simon, Liou, 1990)

Minimum Cut with Constraints

minimize cutsize without explicit size constraints

But where to cut ?



Need to balance sizes

New Objective Functions

$$s(A,B) = \sum_{i \in A} \sum_{j \in B} w_{ij}$$

- Ratio Cut (Hangen & Kahng, 1992)

$$J_{Rcut}(A,B) = \frac{s(A,B)}{|A|} + \frac{s(A,B)}{|B|}$$

- Normalized Cut (Shi & Malik, 2000)

$$d_A = \sum_{i \in A} d_i$$

$$J_{Ncut}(A,B) = \frac{s(A,B)}{d_A} + \frac{s(A,B)}{d_B}$$

$$= \frac{s(A,B)}{s(A,A) + s(A,B)} + \frac{s(A,B)}{s(B,B) + s(A,B)}$$

- Min-Max-Cut (Ding et al, 2001)

$$J_{MMC}(A,B) = \frac{s(A,B)}{s(A,A)} + \frac{s(A,B)}{s(B,B)}$$

Other References

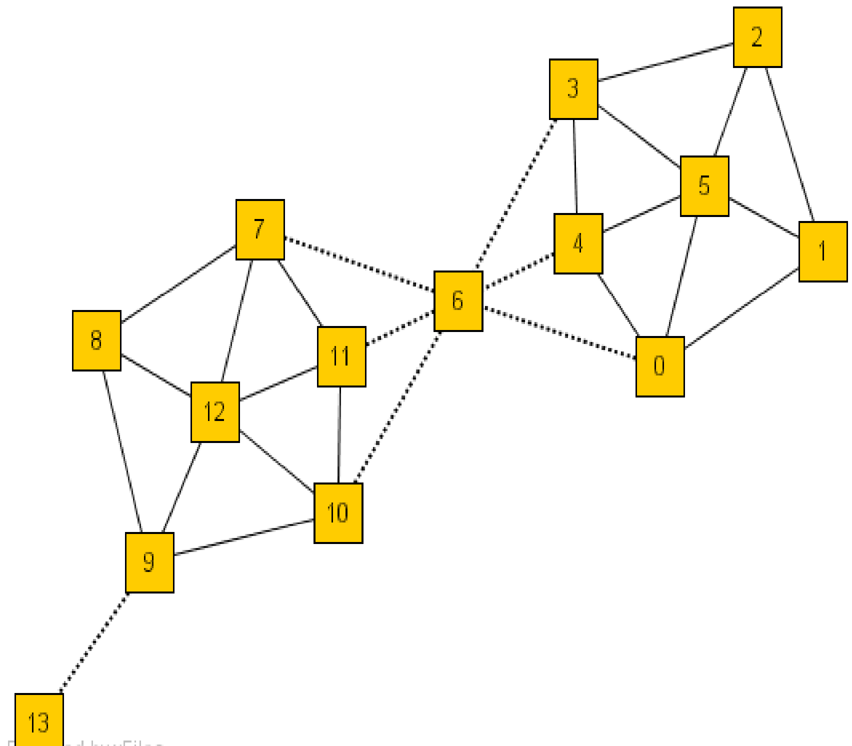
- A Tutorial on Spectral Clustering by U. Luxburg http://www.kyb.mpg.de/fileadmin/user_upload/files/publications/attachments/Luxburg07_tutorial_4488%5B0%5D.pdf

Mining Graph/Network Data

- Graph / Network Data
- Ranking on Graph / Network
- Graph/Network Clustering
- Graph/Network Classification
- Summary

Label Propagation in the Network

- Given a network, some nodes are given labels, can we classify the unlabeled nodes by using link information?
  - E.g., Node 12 belongs to Class 1 and Node 5 Belongs to Class 2



Reference

- Learning from Labeled and Unlabeled Data with Label Propagation
  - By Xiaojin Zhu and Zoubin Ghahramani
  - http://www.cs.cmu.edu/~zhuxj/pub/CMU-CALD-02-107.pdf

Problem Formalization

- Given n nodes
  - l with labels ($Y_1, Y_2, \dots, Y_l$ *are known*)
  - u without labels ($Y_{l+1}, Y_{l+2}, \dots, Y_n$ are unknown)
  - Y *is the* $n \times C$ *label matrix*
    - C is the number of labels (classes)
- The adjacency matrix is W
- The probabilistic transition matrix T
  - $T_{ij} = P(j \to i) = \frac{w_{ij}}{\sum_k w_{kj}}$

The Label Propagation Algorithm

- Step 1: Propagate $Y \leftarrow TY$
- Step 2: Row-normalize Y
  - The summation of the probability of each object belonging to each class is 1
- Step 3: Reset the labels for the labeled nodes. Repeat 1-3 until Y converges

Mining Graph/Network Data

- Graph / Network Data
- Ranking on Graph / Network
- Graph/Network Clustering
- Graph/Network Classification
- Summary

Summary

- Graph / Network Data
  - Adjacency matrix
- Ranking on Graph / Network
  - PageRank
  - Personalized PageRank
- Network Clustering
  - SCAN
  - Spectral clustering
- Network classification
  - Label propagation