Data-driven Intelligent Systems

Lecture 21

Mining Structure from Graphs

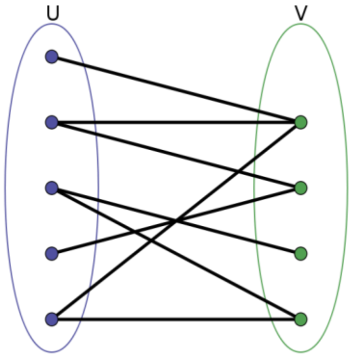
KNOWLEDGE
TECHNOLOGY

http://www.informatik.uni-hamburg.de/WTM/

Overview

- Clustering Graphs
- SCAN
- Label Propagation
- Directed Graph: Webgraph Google

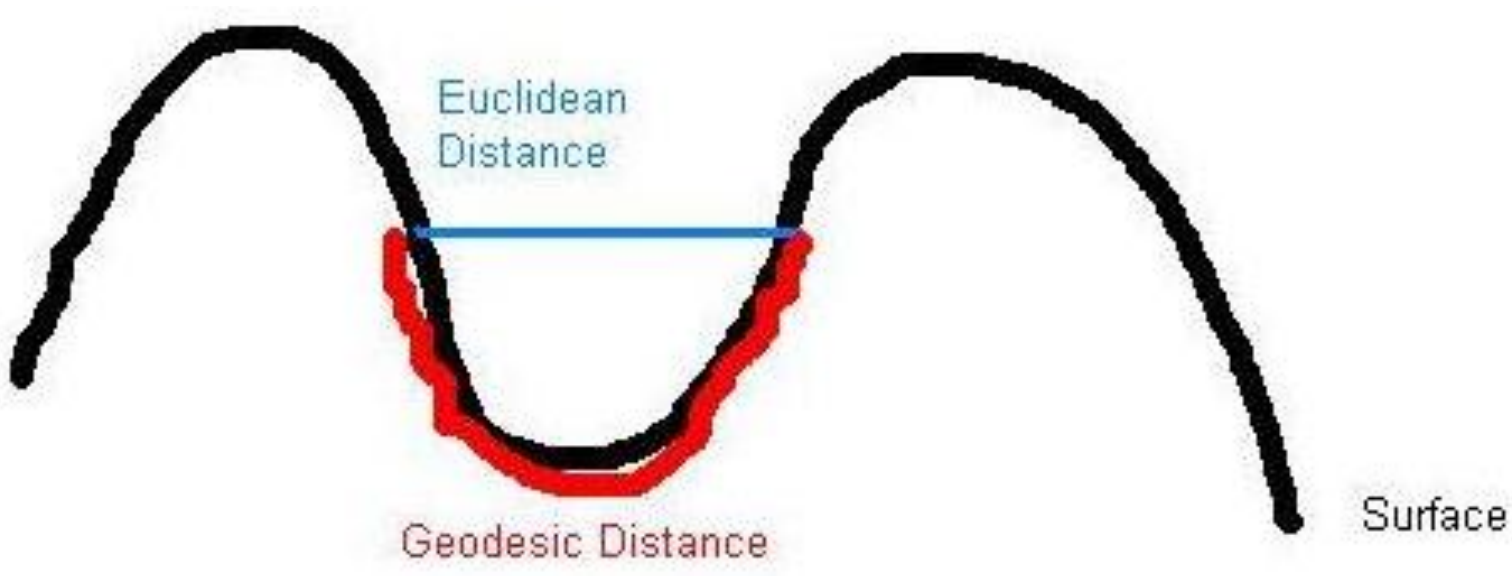
Clustering Graphs and Network Data



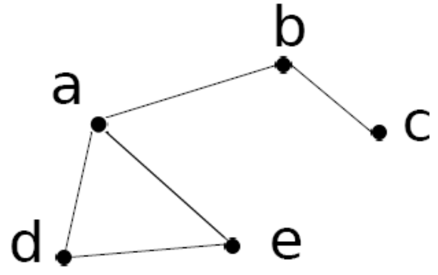
- Applications
  - Bipartite graphs, e.g.:
    - customers and products,
    - authors and conferences, …
  - Items linked in network if frequently purchased together or repeatedly purchased by the same buyer
  - Web search engines, e.g.:
    - click-through graphs, webgraph, …
  - Social networks, friendship/coauthor graphs
- Similarity measures
  - Geodesic distances
  - Structural Similarity

Similarity Measures: Geodesic Distances (1)

- ***Geodesic distance***: distance along curved spaces
- May be approximated by adding many short straight segments, using the Euclidean distance for each of these



Geodesic Distances (2)



- ***Geodesic distance*** (v, u):
  length (i.e., # of edges) of the shortest path between vertices v and u (if not connected, defined as infinite)

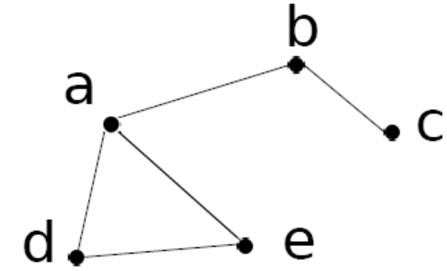
- ***Eccentricity*** of v, eccen(v): The largest geodesic distance between v and any other vertex $u \in V - \{v\}$.
  - E.g.,

    eccen(a) = eccen(b) = 2;

    eccen(c) = eccen(d) = eccen(e) = 3

- A ***peripheral vertex*** is a vertex that achieves the diameter.
  - **e.g.** vertices c, d, and e

Geodesic Distances (3)



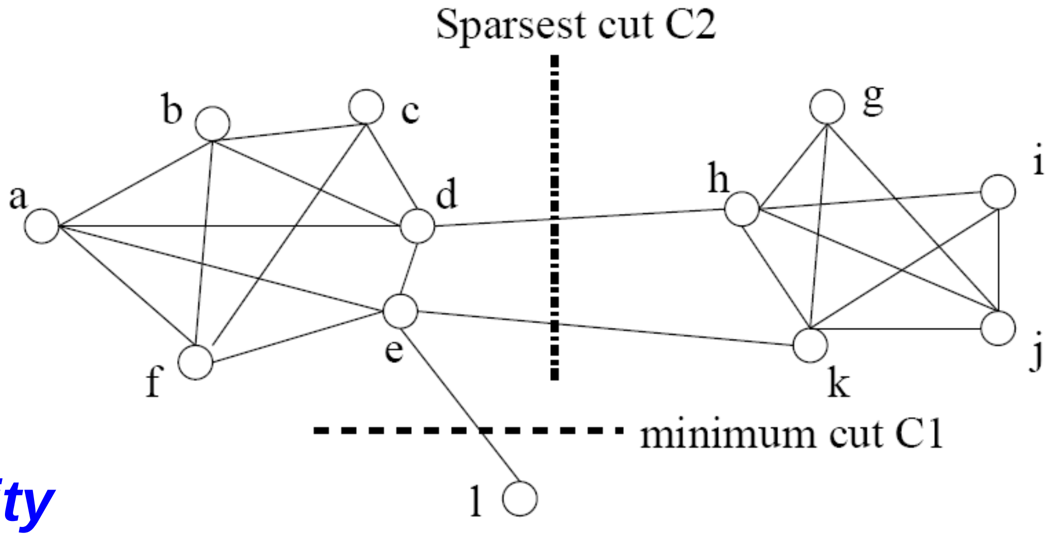
- ***Radius*** of a graph G:
  The minimum eccentricity of all vertices,
  i.e., the distance between the
  "most central point" and the "farthest border"
  - $r = \min_{v \in V} \text{eccen}(v)$
  - **E.g.**, radius (G) = 2

- ***Diameter*** of graph G: The maximum eccentricity of all vertices, i.e., the largest distance between any pair of vertices in G
  - $d = \max_{v \in V} \text{eccen}(v)$
  - **E.g.**, diameter (G) = 3

Graph Clustering: Sparsest Cut (1)

Undirected graph *G* = (*V*,*E*).

- The ***cut set*** of a cut is the set of edges $\{(u, v) \in E \mid u \in S, v \in T\}$ where nodes *u* and *v* are in the two partitions *S* and *T*



- ***Size*** of the cut:
  # of edges in the cut set
- Min-cut (e.g., C_1)
  is not a good partition
- A better measure: ***Sparsity***

$$\Phi = \frac{\text{the size of the cut}}{\min\{|S|, |T|\}}$$

Graph Clustering: Sparsest Cut (2)

- A cut is ***sparsest*** if its sparsity is not greater than that of any other cut
- **Ex**. Cut C2 = ({*a, b, c, d, e, f, l*}, {*g, h, i, j, k*}) is the sparsest cut
- For k clusters, the ***modularity*** of a clustering assesses the quality of the clustering
- The *modularity* of a clustering of a graph is the difference between the fraction of all edges *within* individual clusters *(this should be large)* and the fraction of all edges *between* different clusters *(this should be small)*:

$$Q = \sum_{i=1}^{k} \left(\frac{l_i}{|E|} - \left(\frac{d_i}{2|E|} \right)^2 \right)$$

l_i: # edges between vertices *within* i-th cluster
d_i: # *all* edges connecting to vertices in i-th cluster

- The optimal clustering of graphs maximizes the modularity

Graph Clustering: Challenges of Finding Good Cuts

- High computational cost
  - Many graph cut problems are computationally expensive
  - The sparsest cut problem is NP-hard
  - Need a tradeoff between efficiency/scalability and quality
- Sophisticated graphs
  - May involve weights
  - Directed graphs may contain cycles (undirected graphs, too)
- 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 will also be sparse

Two Approaches for Graph Clustering

1. Methods specifically designed for clustering graphs
   - Search the graph to find well-connected components as clusters
   - Ex. SCAN: Structural Clustering Algorithm for Networks
   - Ex. Label propagation (Chinese Whispers)

2. Using generic clustering methods for high-dimensional data
   - Extract a similarity/affinity matrix $A \in R^{nxn}$ (*n* = number of nodes) from a graph using a similarity measure, for example:
     - similarity = 1, if nodes connected, else 0
     - similarity = Gauß(distance between nodes)
   - Discover clusters on the similarity matrix by a generic method, e.g. spectral clustering

     → approximate optimal graph cut solutions

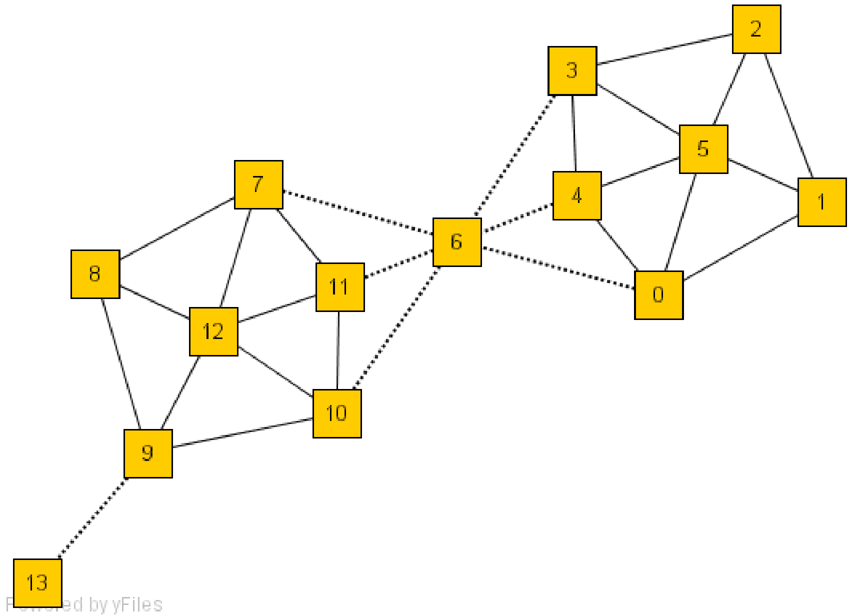
applicable also to non-graph data

Overview

- Clustering Graphs
- SCAN
- Label Propagation
- Directed Graph: Webgraph Google

SCAN: Density-Based Clustering of Networks

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

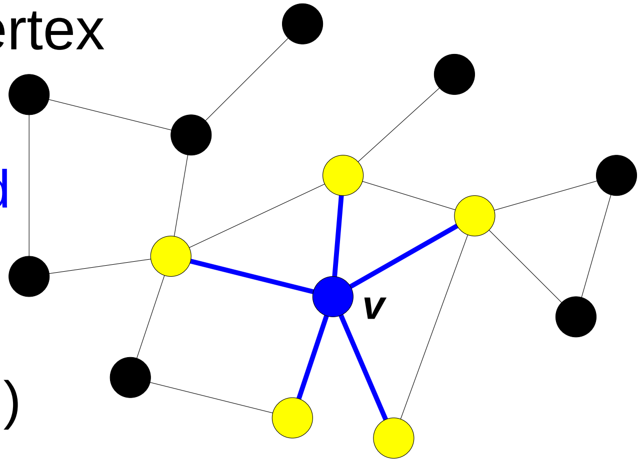


Application

- Given: information of who associates with whom
- Identify clusters of individuals with common interests or special relationships (families, cliques, terrorist cells) …

A Social Network Inspired Model

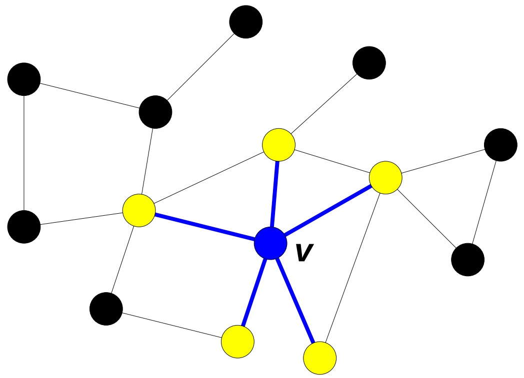
- Characteristics:
  - Individuals in a tight social group, or ***clique***, know many of the same people, regardless of the size of the group
  - Individuals who are ***hubs*** know many people in different groups but belong to no single group. E.g., politicians bridge multiple groups
  - Individuals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



Structure Similarity

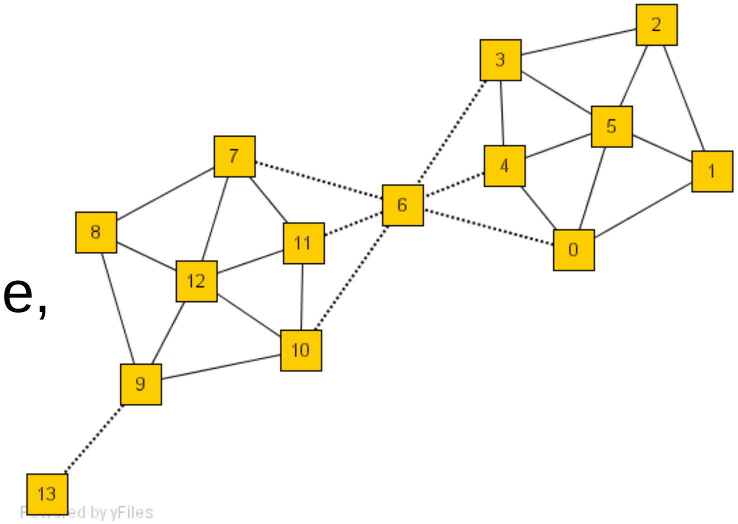
- The desired characteristics tend to be captured by a measure we call Structural Similarity

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



  - $0 \leq \sigma \leq 1$

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



Structural Connectivity

- ε-neighborhood: $N_\varepsilon(v) = \{w \in \Gamma(v) \mid \sigma(v,w) \geq \varepsilon\}$
- Vertex is a core: $CORE_{\varepsilon,\mu}(v) \Leftrightarrow |N_\varepsilon(v)| \geq \mu$ μ integer

  we will let structures grow starting from the core
- Direct structure reachable:

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

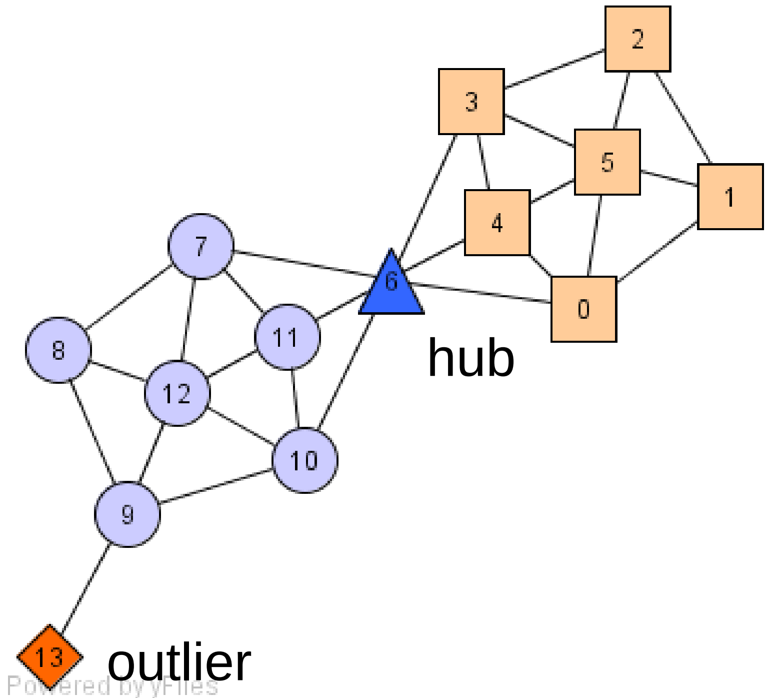
- Structure reachable: $REACH_{\varepsilon,\mu}(v,w)$ transitive closure of direct structure reachability
- Structure connected:

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

Xu, Yuruk, Feng, Schweiger (SIGKDD'07) "SCAN: A Structural Clustering Algorithm for Networks". See also: Ester, Kriegel, Sander, Xu (KDD'96) "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases".

Structure-Connected Clusters

- Define a **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



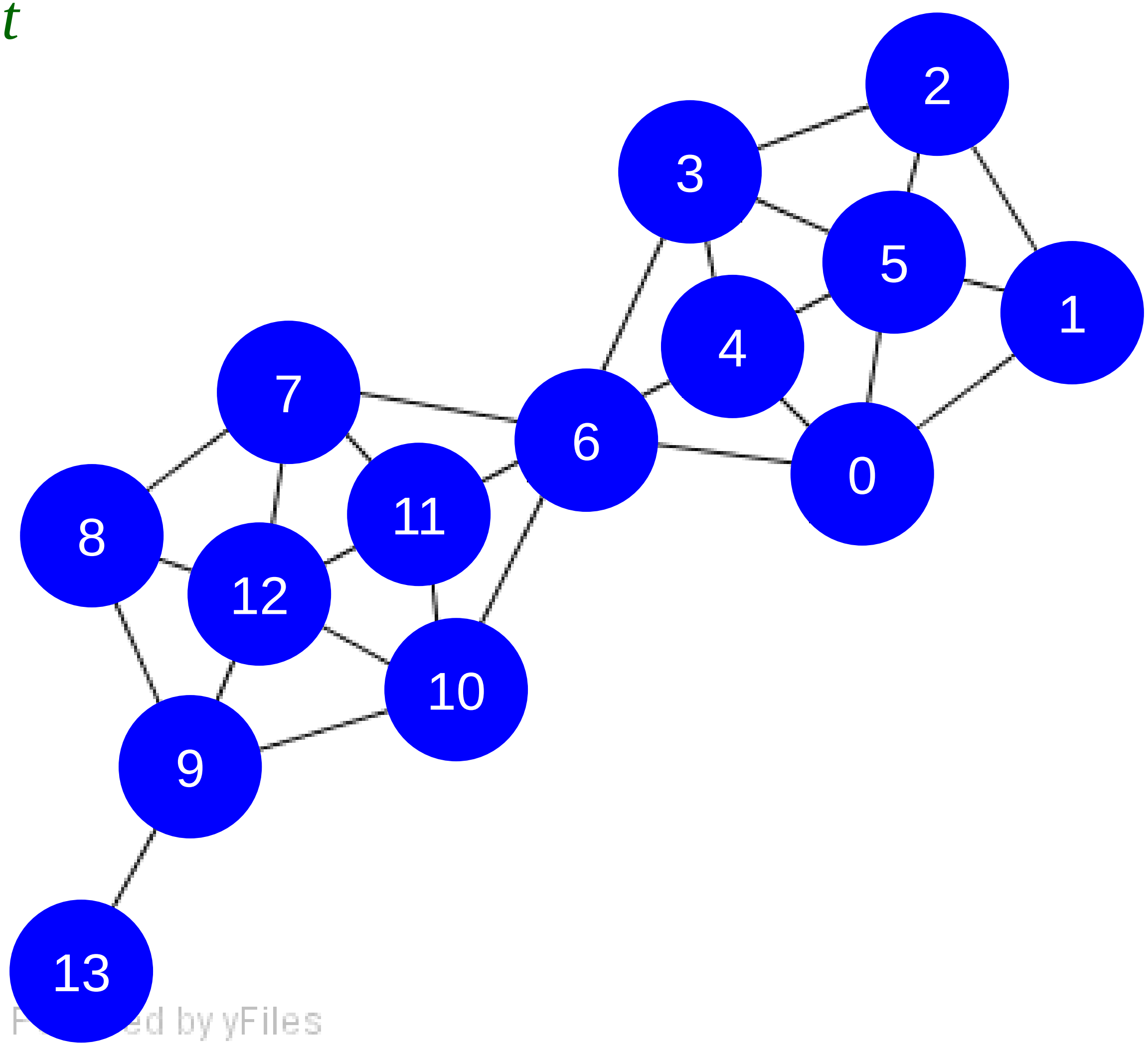
SCAN Algorithm

required neighbours that vertex is core

$\mu = 2$

$\varepsilon = 0.7$

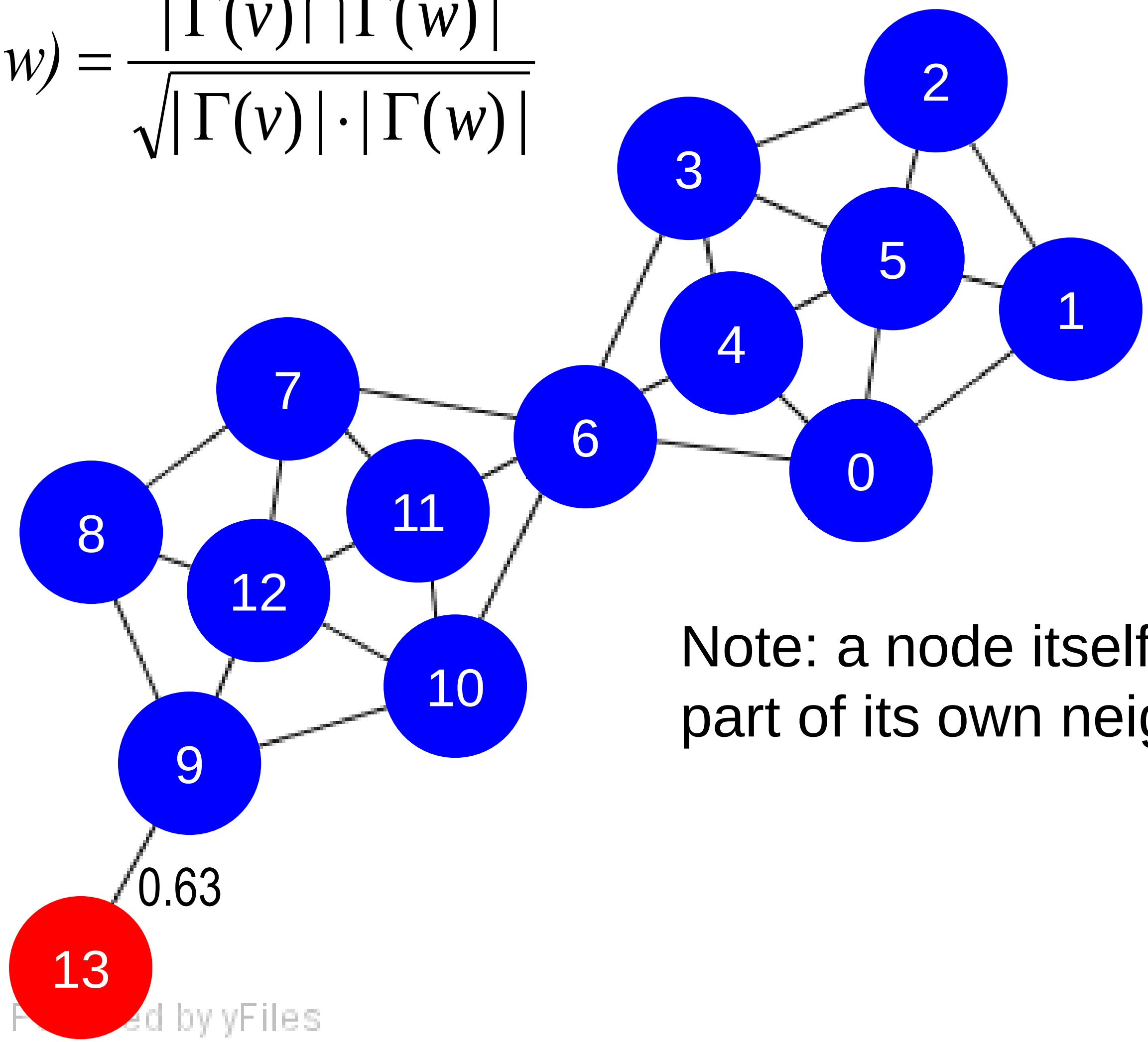
required similarity



SCAN Algorithm

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

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



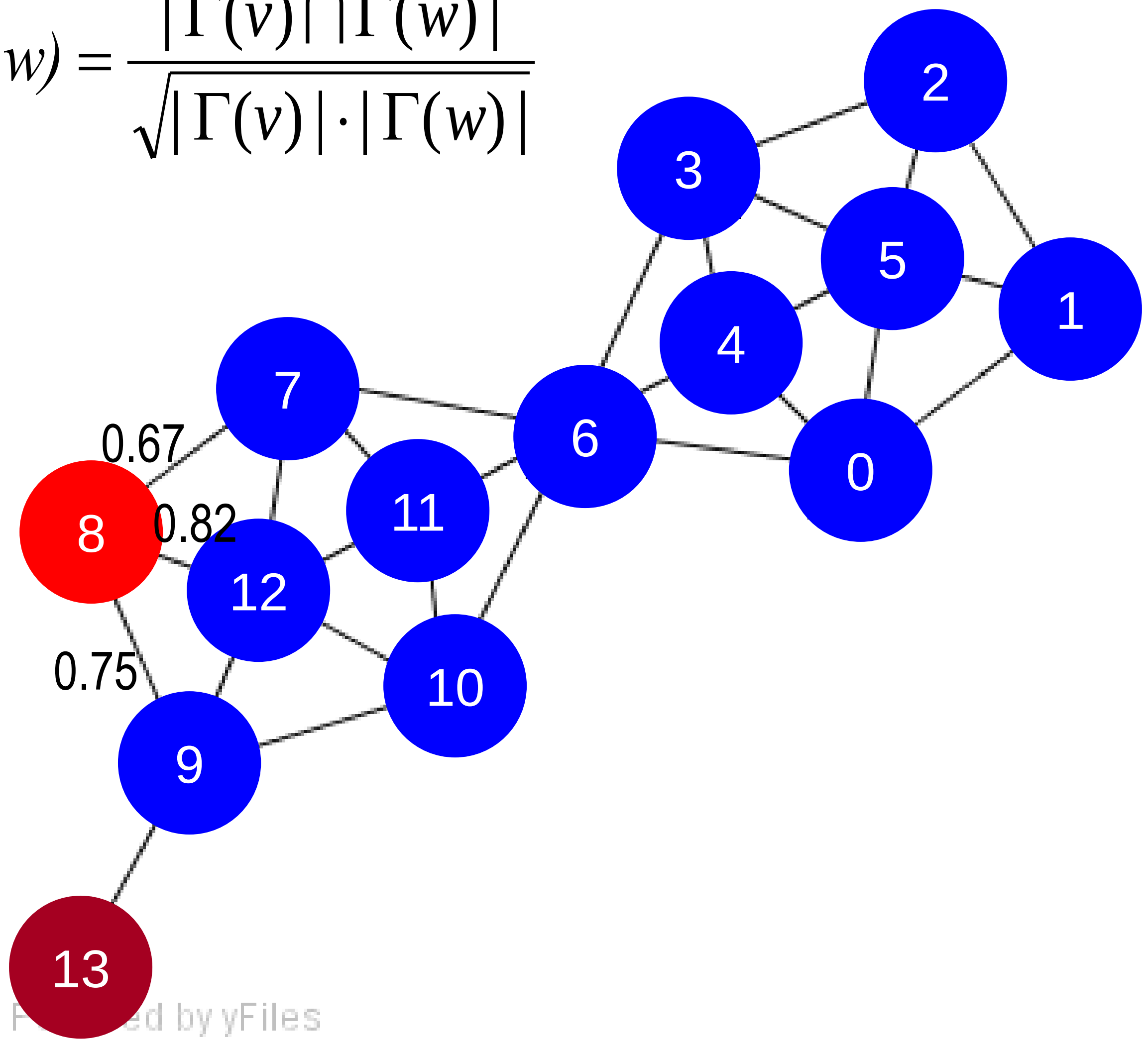
Note: a node itself counts as part of its own neighborhood

SCAN Algorithm

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

$\mu = 2$

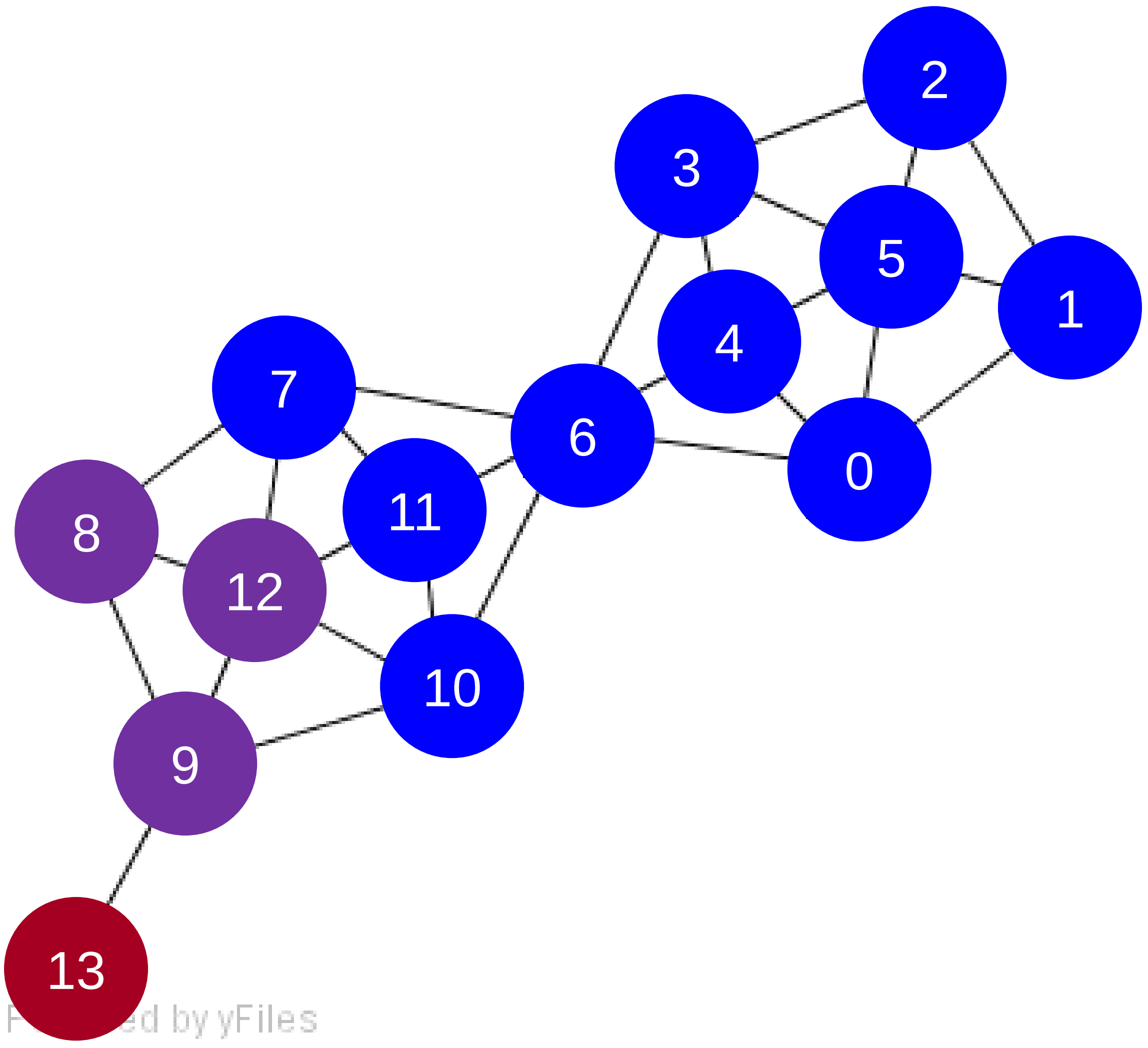
$\varepsilon = 0.7$



SCAN Algorithm

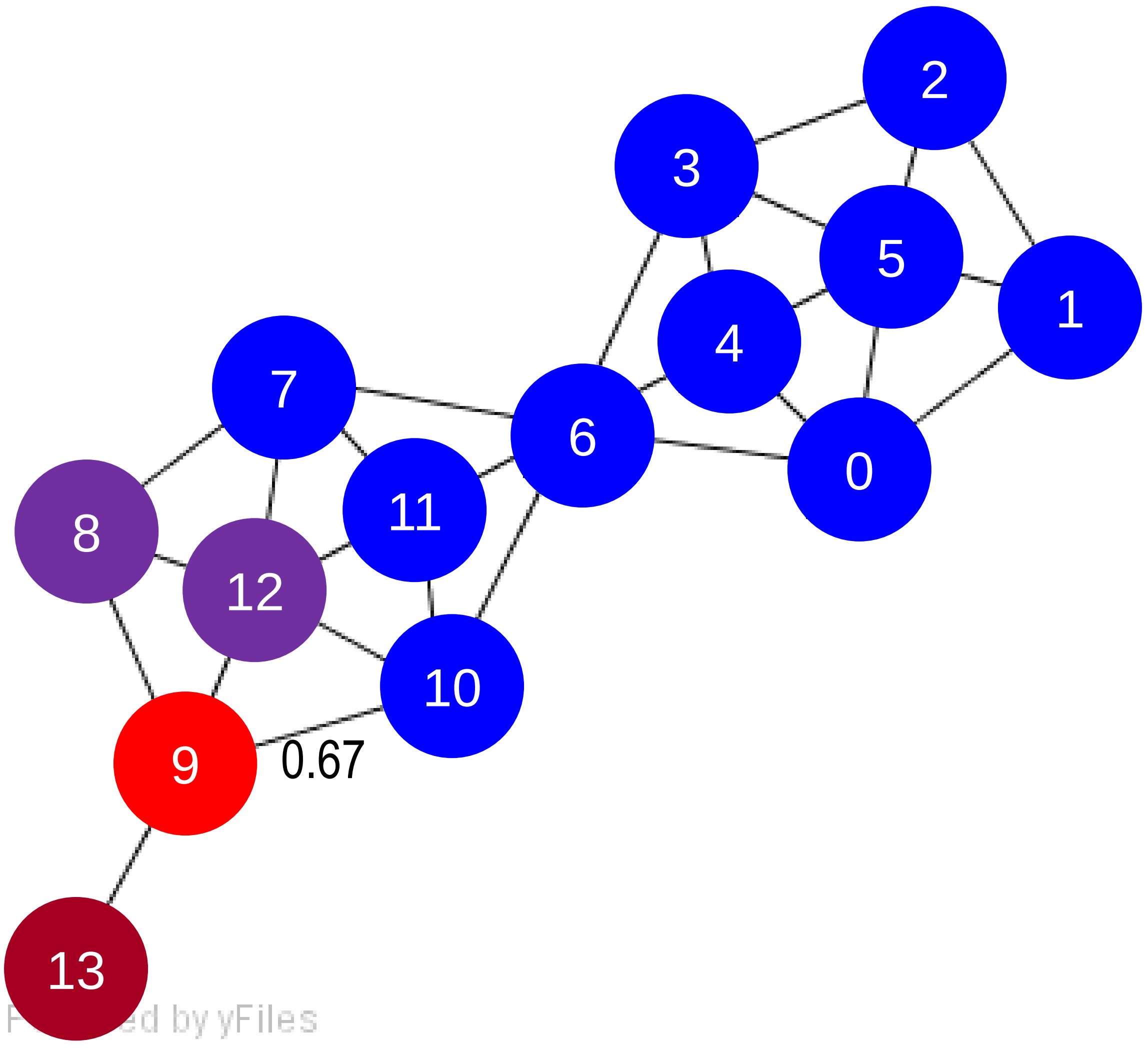
$\mu = 2$
$\varepsilon = 0.7$

purple: a confirmed clique



SCAN Algorithm

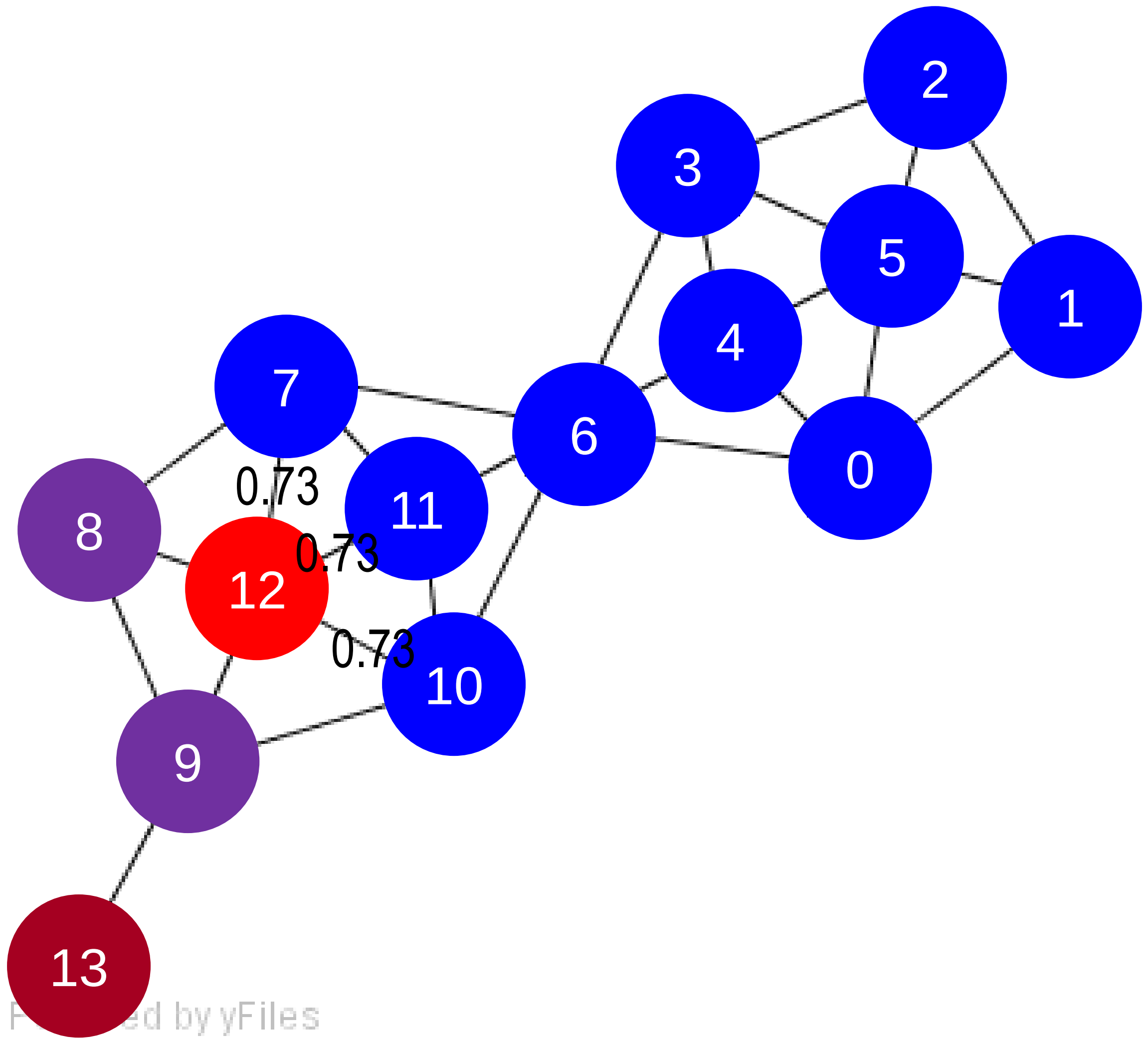
$\mu = 2$
$\varepsilon = 0.7$



SCAN Algorithm

$\mu = 2$

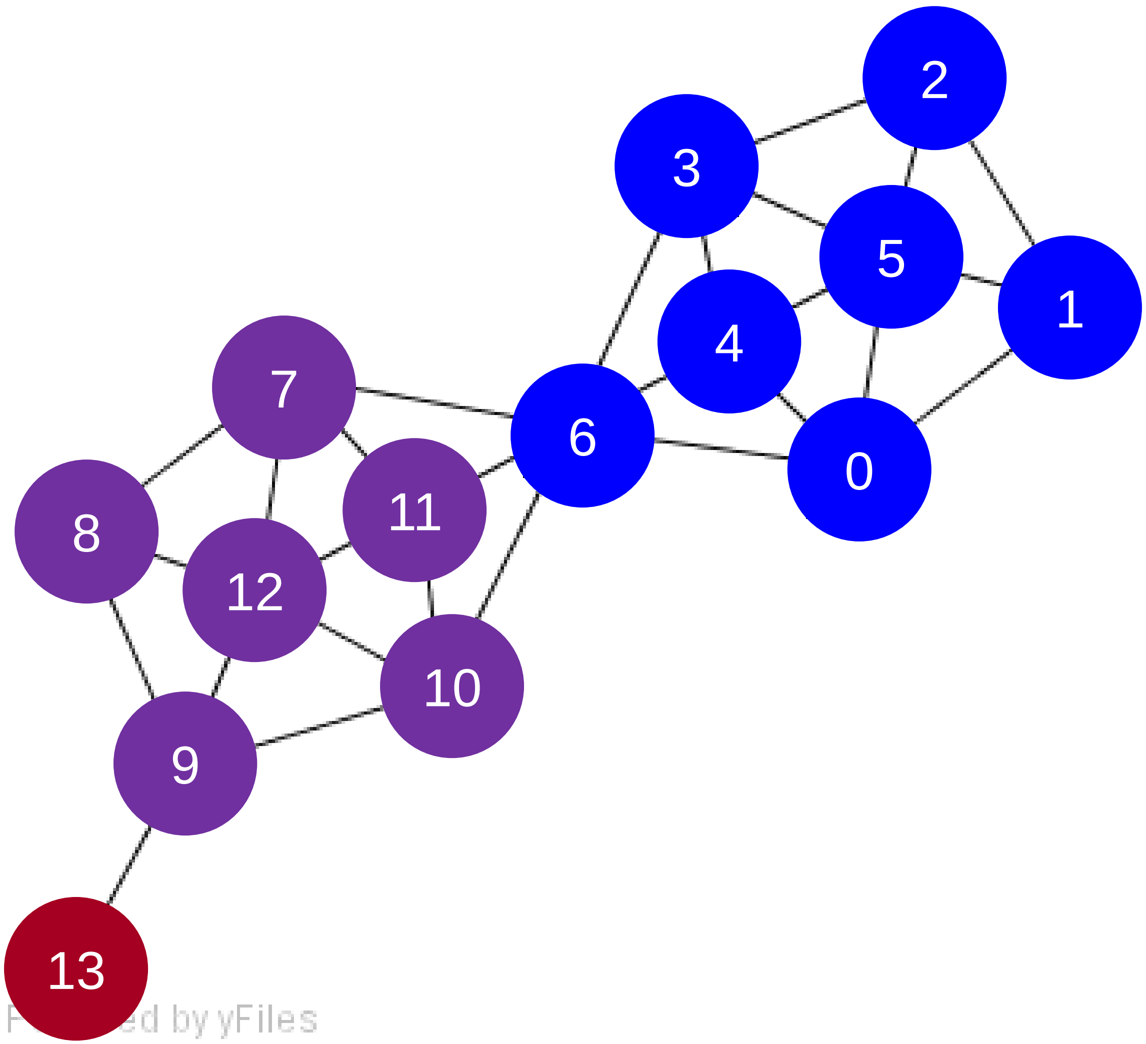
$\varepsilon = 0.7$



SCAN Algorithm

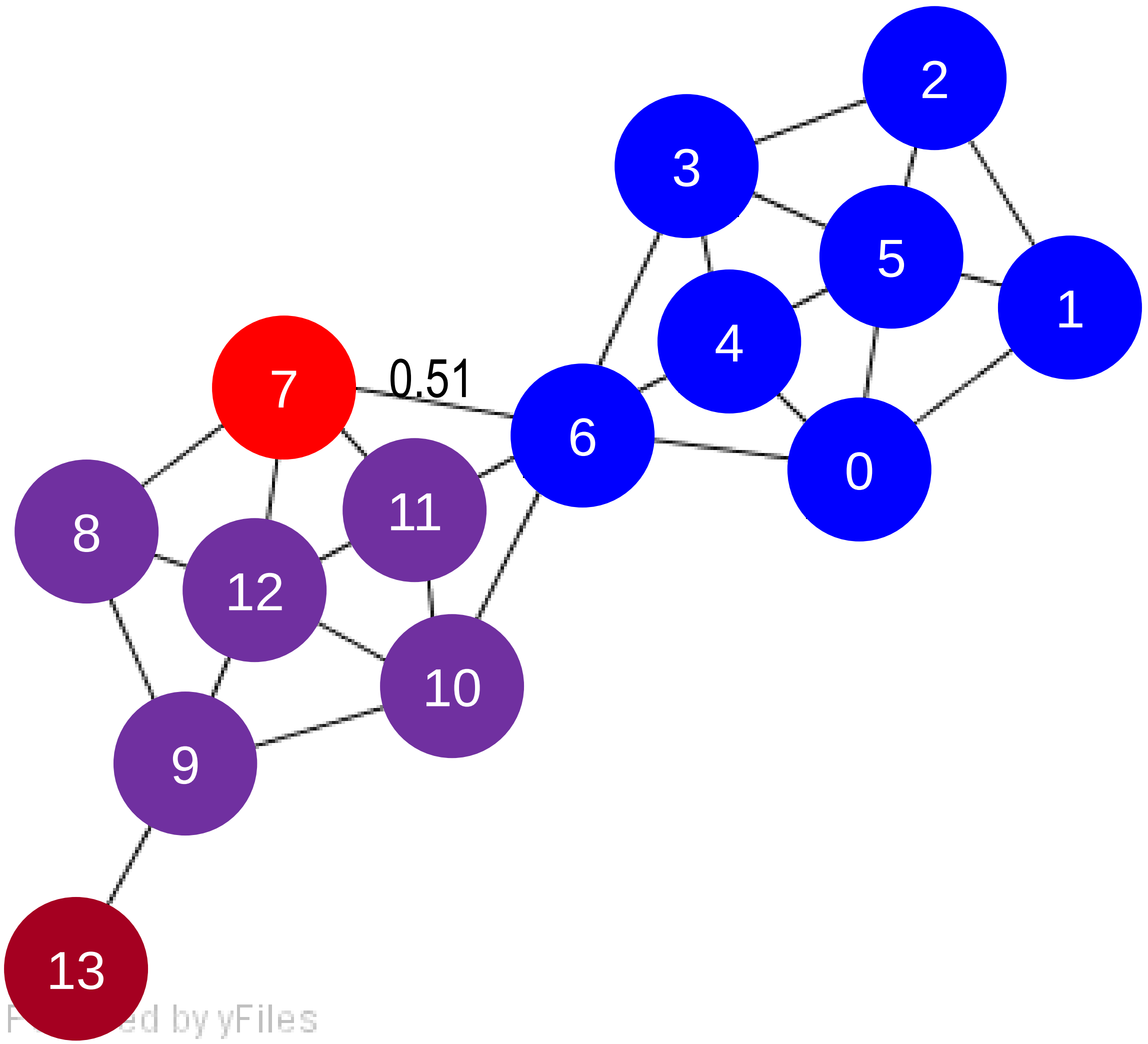
$\mu = 2$

$\varepsilon = 0.7$



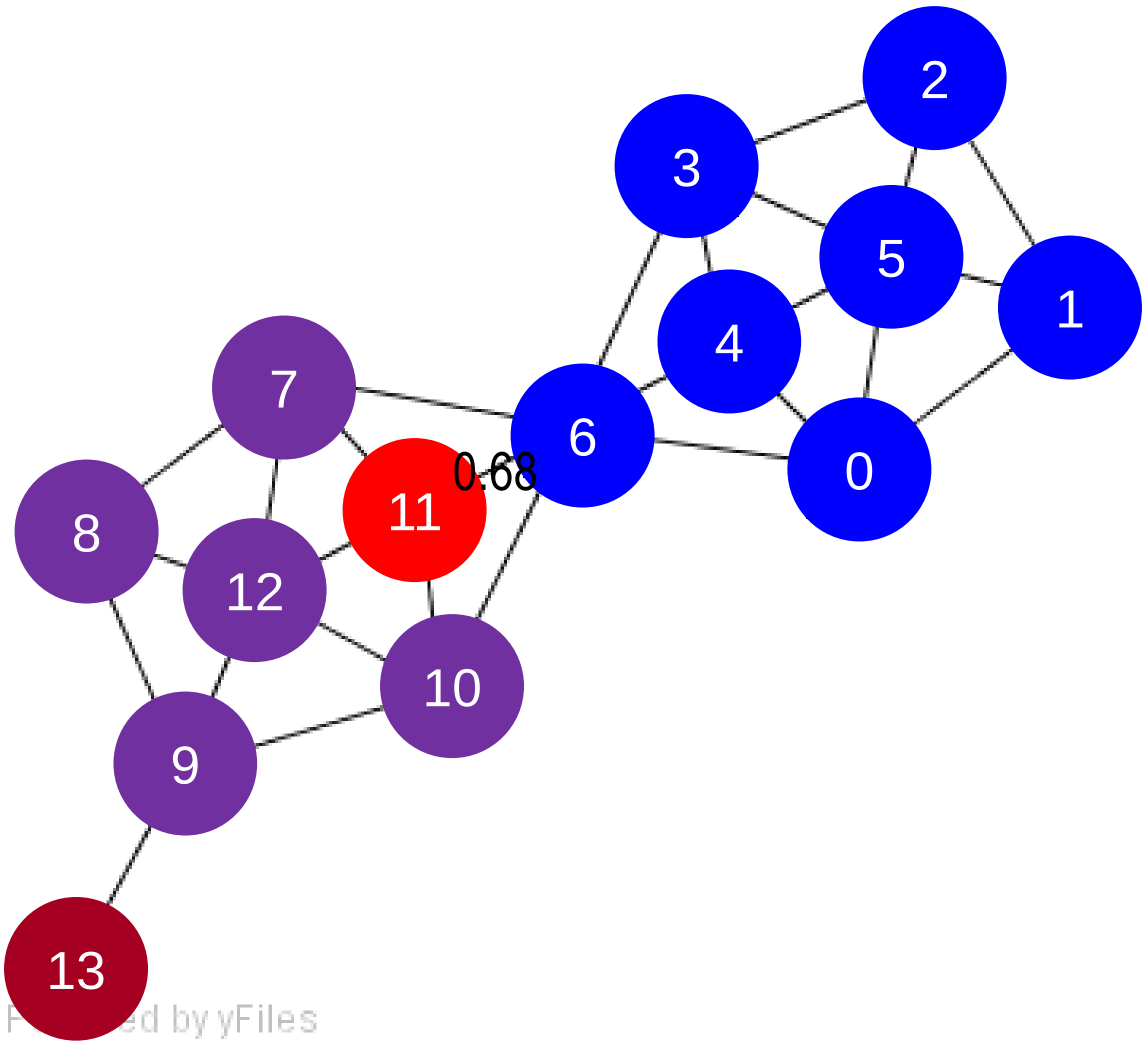
SCAN Algorithm

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



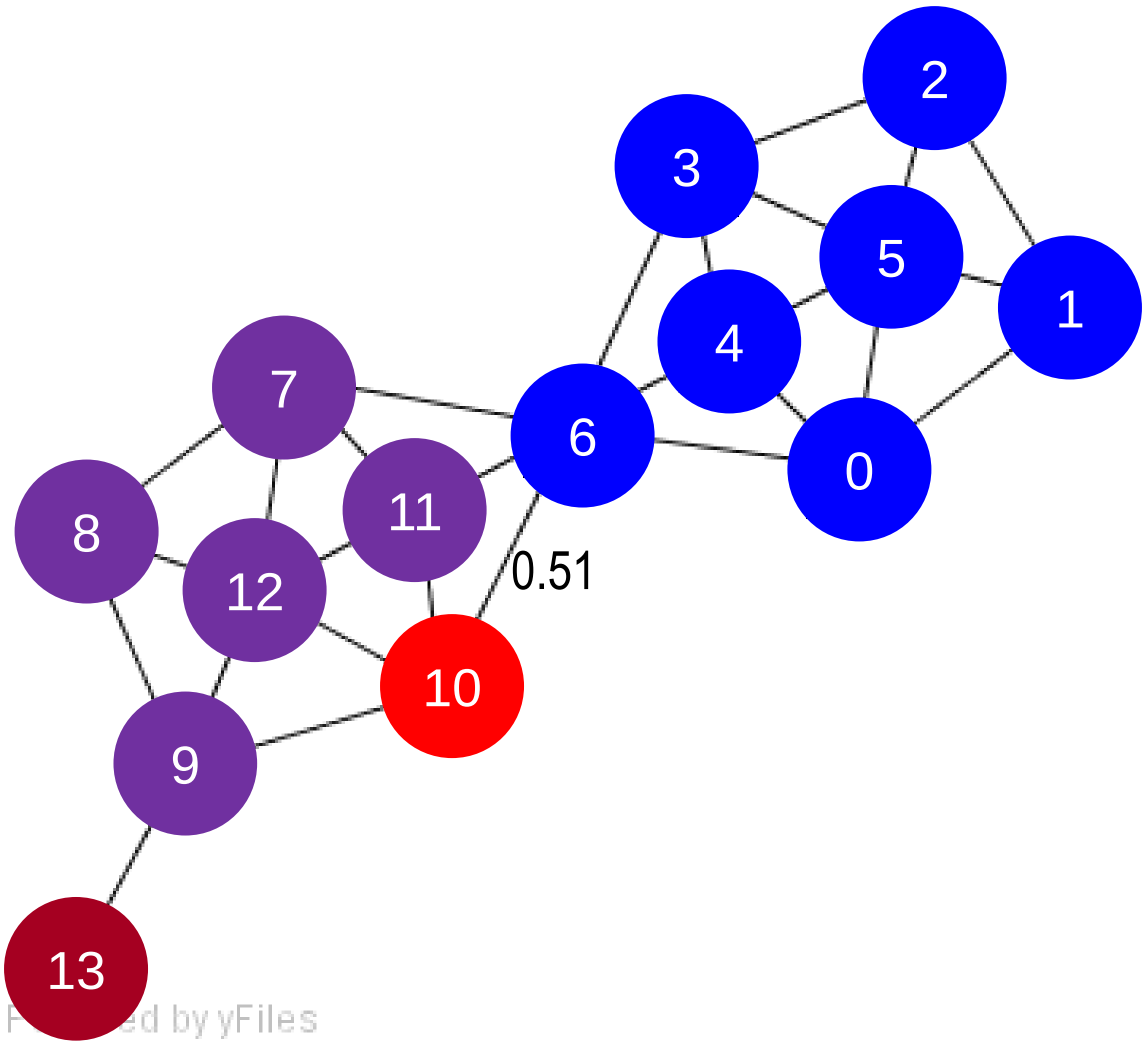
SCAN Algorithm

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



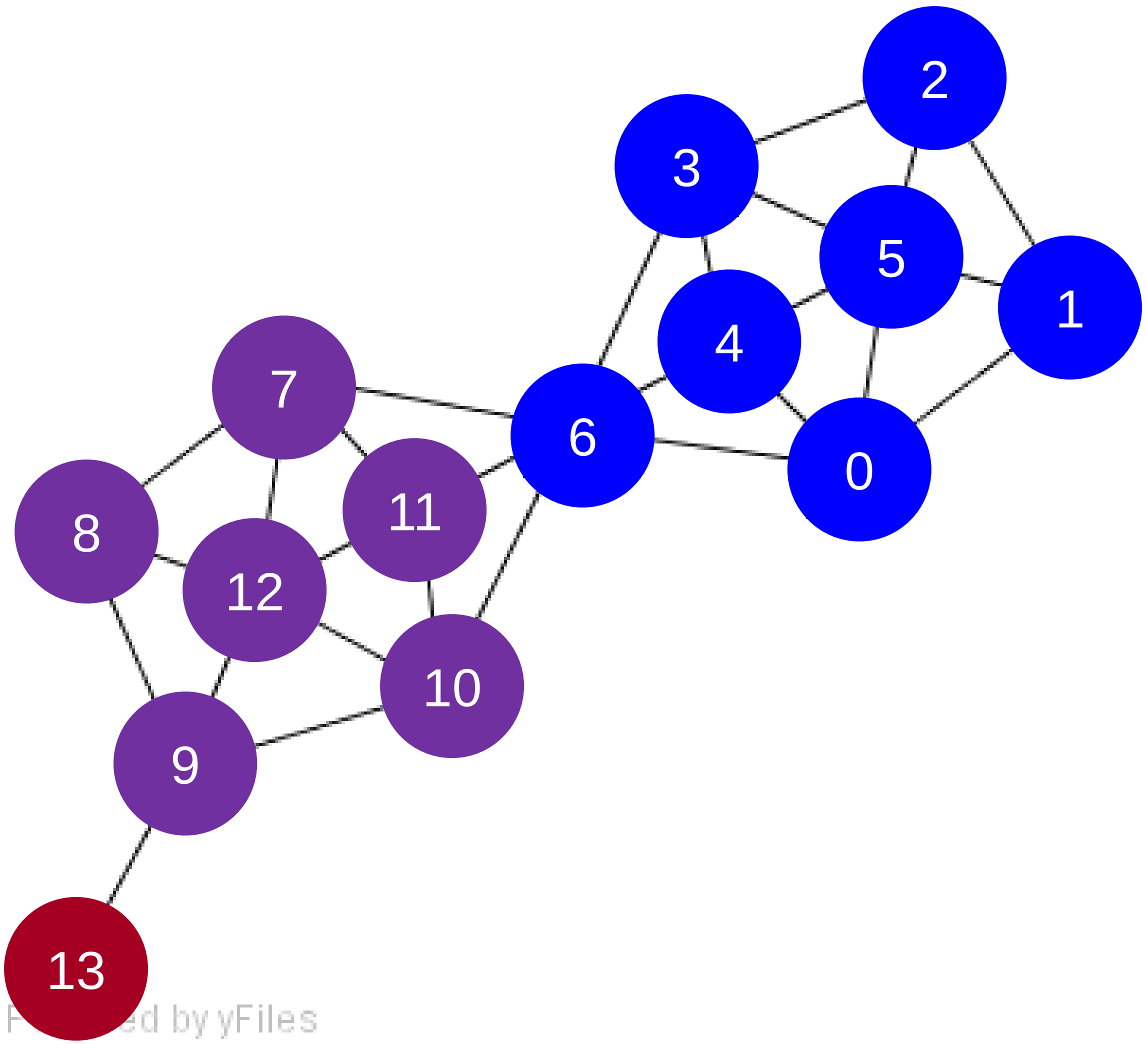
SCAN Algorithm

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



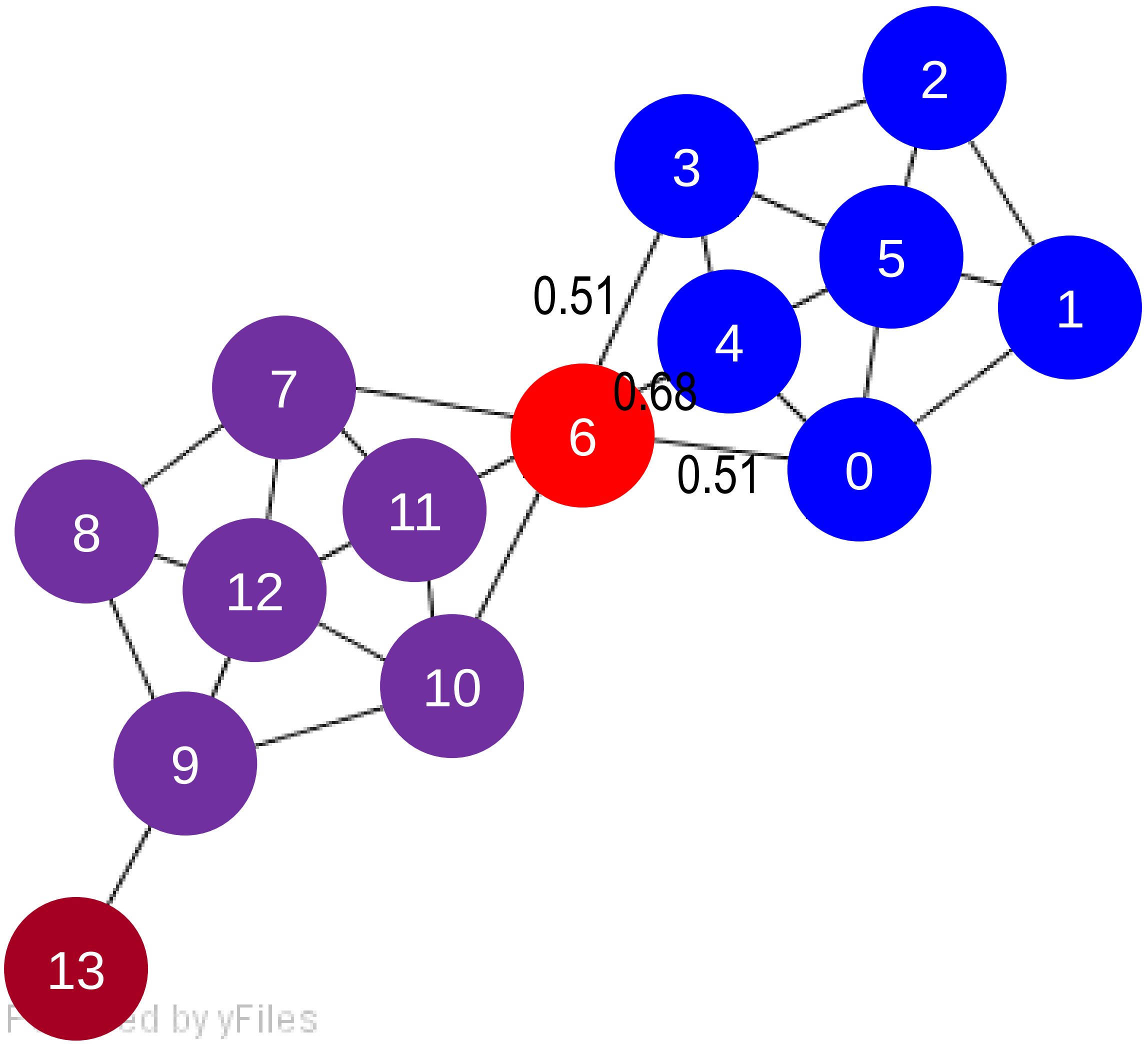
SCAN Algorithm

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



SCAN Algorithm

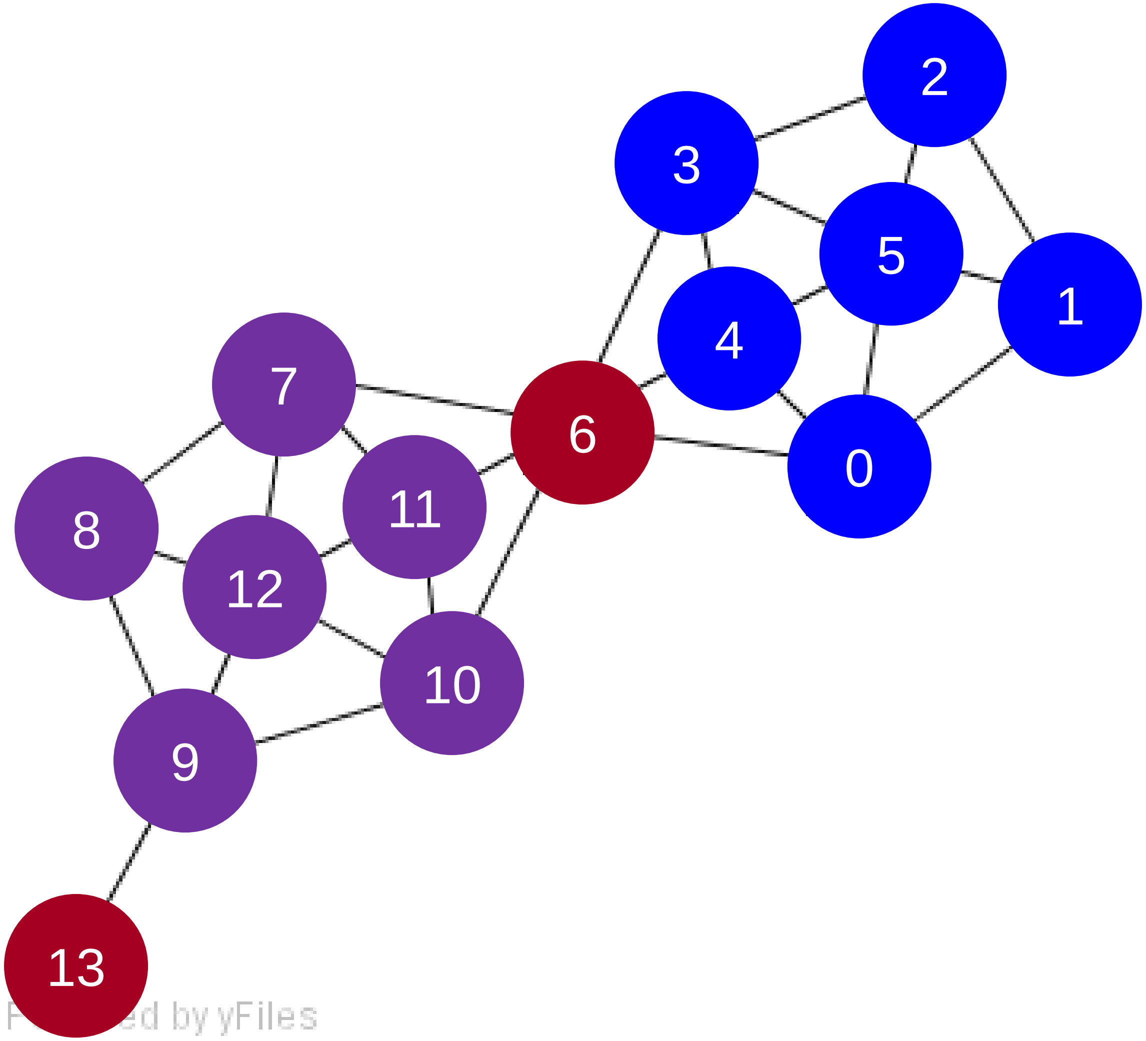
$\mu = 2$
$\varepsilon = 0.7$



SCAN Algorithm

$\mu = 2$

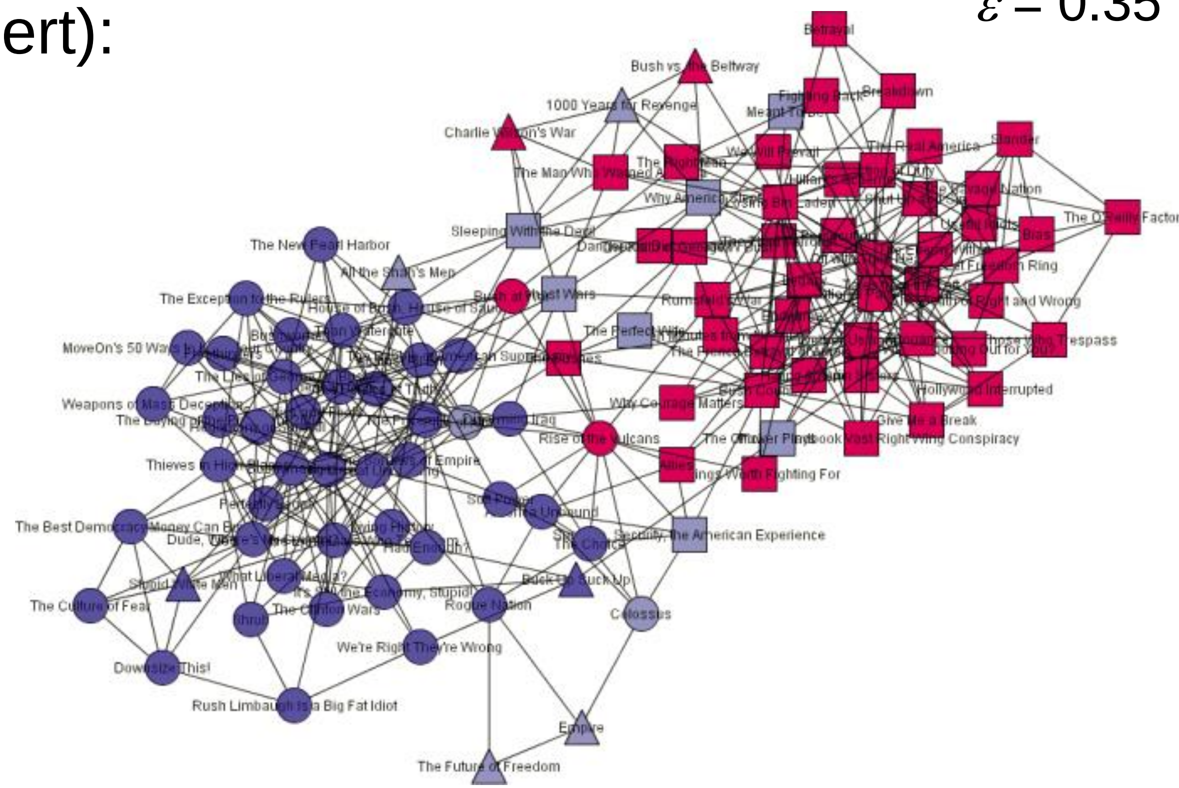
$\varepsilon = 0.7$



SCAN – Political Book Graph

- Data: political books sold on Amazon.
- Books are linked if co-purchased by a customer.
- True labels (via expert):
  - Conservative: red
  - Neutral: grey
  - Liberal: blue
- SCAN successfully finds three clusters:
  - Square
  - Triangle
  - Circle

$\mu = 2$
$\varepsilon = 0.35$



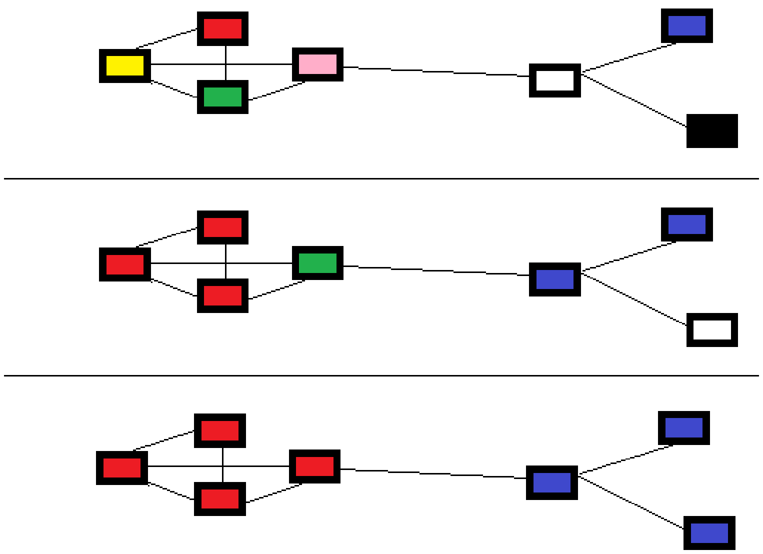
Xu et al. (2007) SCAN: A Structural Clustering Algorithm for Networks. SIGKDD 2017

Overview

- Clustering Graphs
- SCAN
- Label Propagation
- Directed Graph: Webgraph Google

Label Propagation: Chinese Whispers

- Name CW after the children`s game *(German: “Stille Post“)*
- Idea: nodes become community (cluster) members
  - nodes send the same type of information to each other



- Works on undirected graph
- Links can be binary or weighted

Chris Biemann. Chinese Whispers - an Efficient Graph Clustering Algorithm and its Applications to Natural Language Processing Problems, TextGraphs, HLT-NAACL, 2006

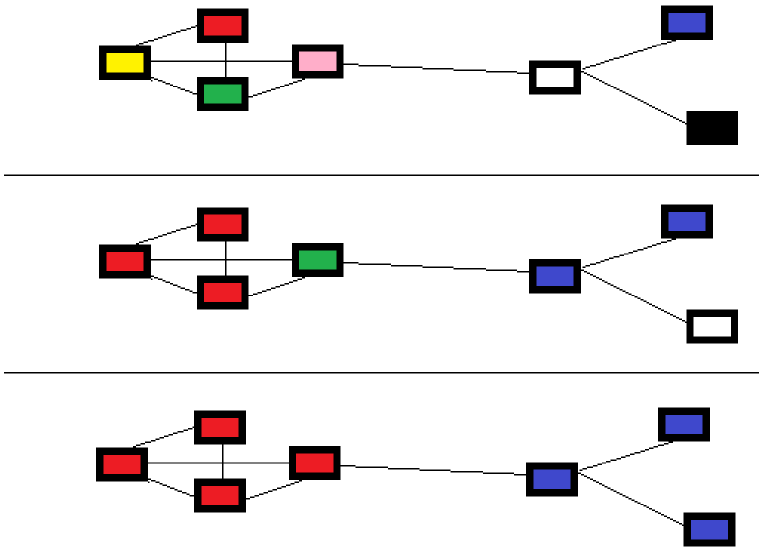
See also: Raghavan, Albert, Kumara. Near linear time algorithm to detect community structures in large-scale networks. Phys. Rev. E, 2007.

Chinese Whispers – Algorithm

- Init: all nodes are assigned to a random class.
  - The number of initial classes equals the number of nodes.
- Repeat:
  - Select a random node.
  - Node inherits the class whose sum of edge weights to it is maximal
    - in the case of multiple strongest classes, choose randomly
- Stop when the process converges
  - Few nodes may not converge … then stop at a predetermined number of iterations.
- The emerged classes represent the clusters of the network.
  - The algorithm chooses the number of clusters on its own.

Chinese Whispers – Properties

- Algorithm does random node selection and hard assignments
  - Different solutions at each run
  - A problem for small networks
- Execution time is linear
  - Good for large networks
  - Particular effective
    if the network has the
    small world property



Chinese Whispers – Energy Function

- One can write down an energy function, which is a scalar value E that depends on the cluster assignments $\{c_i\}$ as:

$$\mathrm{E} = -\frac{1}{2} \sum_{i,j} w_{ij}\, \delta_{c_i c_j} = -\sum_{i<j} w_{ij} \delta_{c_i c_j}$$

  where c_i is the class assigned to node i

  and $\delta_{c_i c_j}$ = 1, if $c_i = c_j$ and $\delta_{c_i c_j}$ = 0, if $c_i \neq c_j$

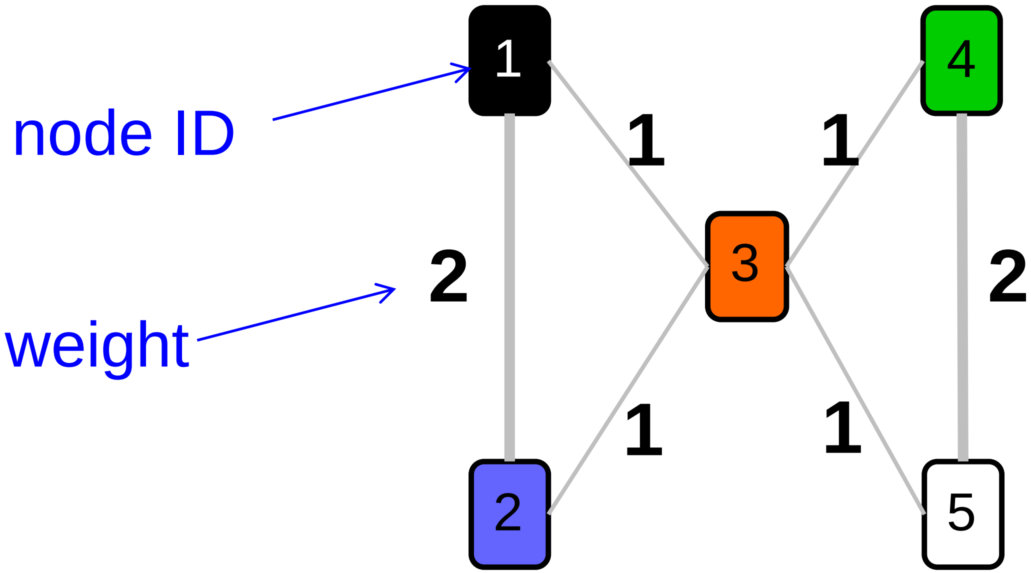
  → E is obtained by summing up all weights of edges that connect two nodes of same class.

- An update of a node\`s class using CW will reduce E (or keep it constant), but will never increase E

  → CW converges to a local minimum of E

Weber, Wermter, Biemann et al. An Energy Function for the Cluster Assignments of the Chinese Whispers Algorithm, 2019, in preparation.

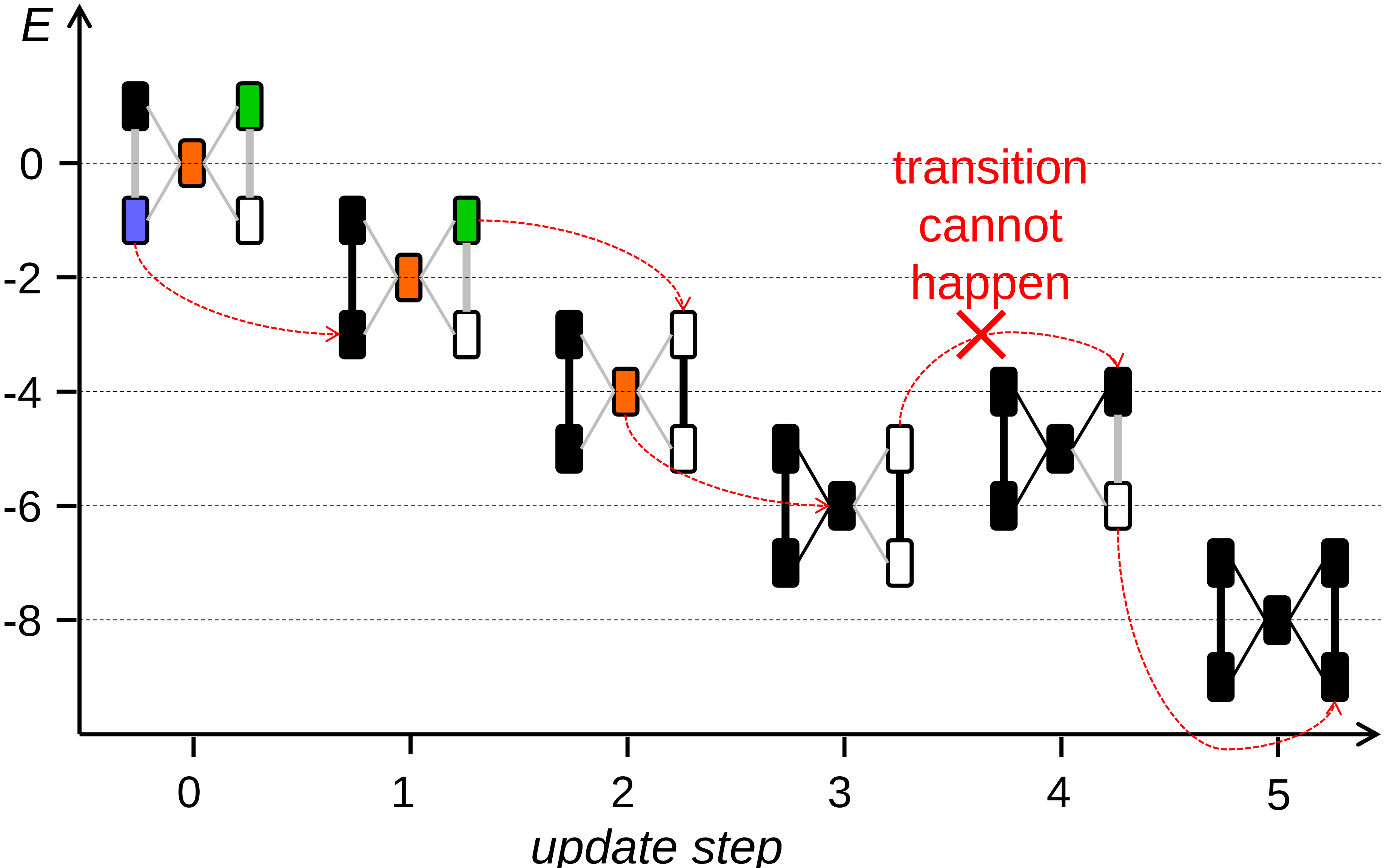
Chinese Whispers – Example



colors denote classes

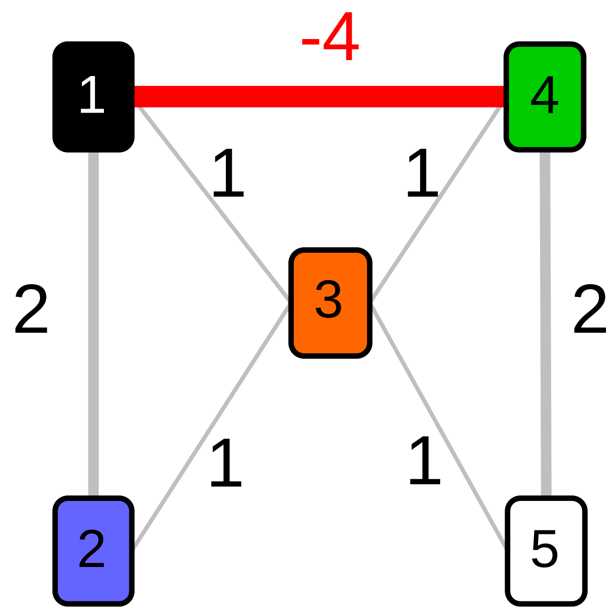
See also: Tibély, Kertész. On the equivalence of the label propagation method of community detection and a Potts model approach. Physica A: Stat Mech Appl. 2008.

Chinese Whispers – Example



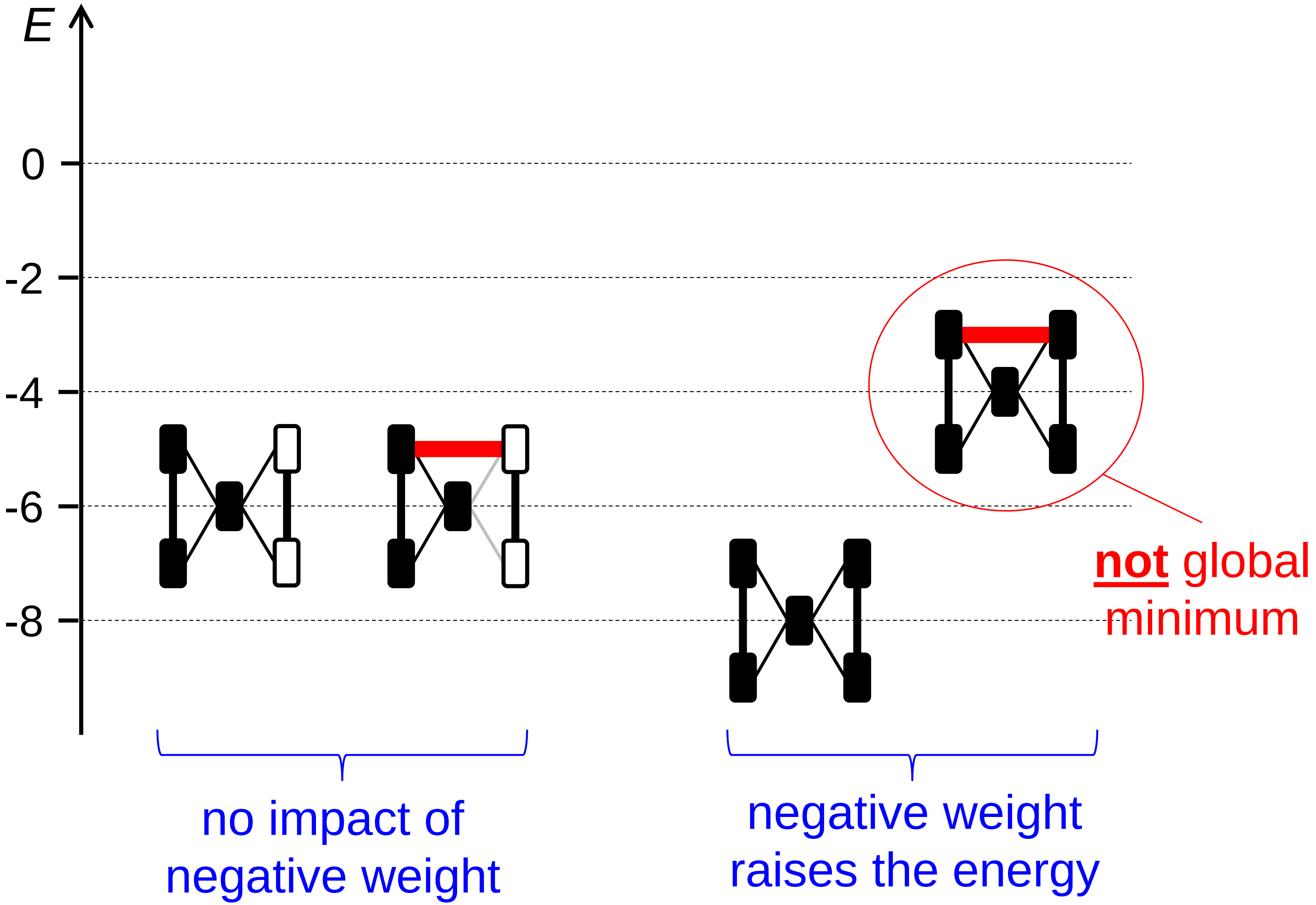
Chinese Whispers – Example

- If all nodes assigned to one class ↔ global energy minimum
  - ***Mostly***, this does not happen, due to initial random classes
- To “force“ units to belong to different classes, one could introduce negative weights

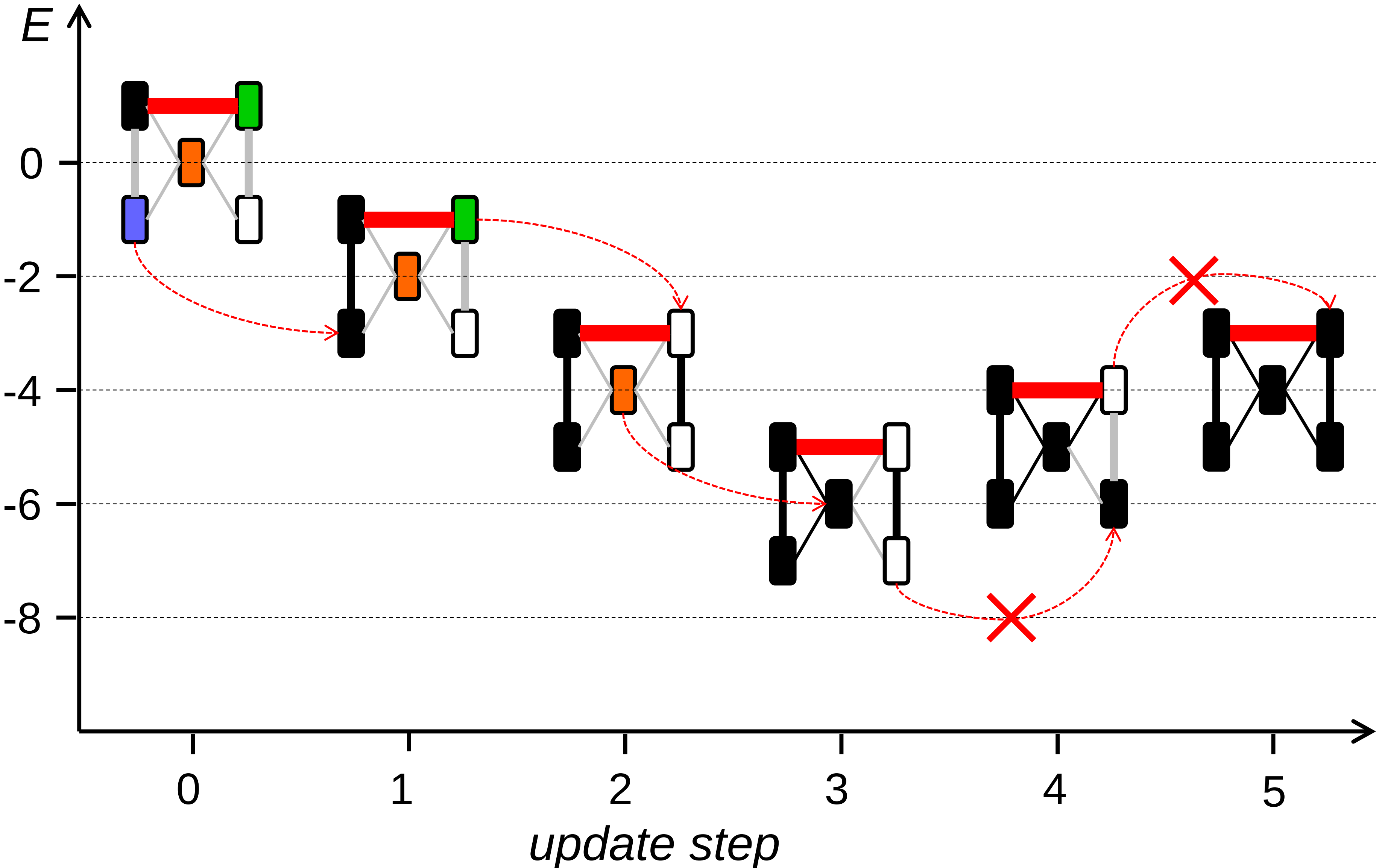


→ *constraint-based clustering*

Chinese Whispers – Example



Chinese Whispers – Example



Chinese Whispers – Energy Function

CW minimizes an energy, like a Hopfield network does

Chinese Whispers		Hopfield network
k clusters (*k* unknown)	—	two activation states (+1, -1)
Assign classes to minimize the energy	—	Update activities to minimize the energy
Graph typically sparse; weights are given	—	Graph fully connected; weights by Hebbian learning
Desired: unique way to form *k* clusters (but global minimum may not be desirable)	—	May find *n* ways (# patterns) to separate activation states (closest pattern to start pattern desired, but not global minimum)

Chinese Whispers – Energy Function

Minimizing an energy of CW is very **different** from minimizing an error function of supervised learning (e.g. neural network)

Chinese Whispers

- Graph and its weights are given
- Energy *E* depends on weights and assignments
- Energy *E* minimized via class assignments
- Global minimum may not be desirable

Supervised Learning

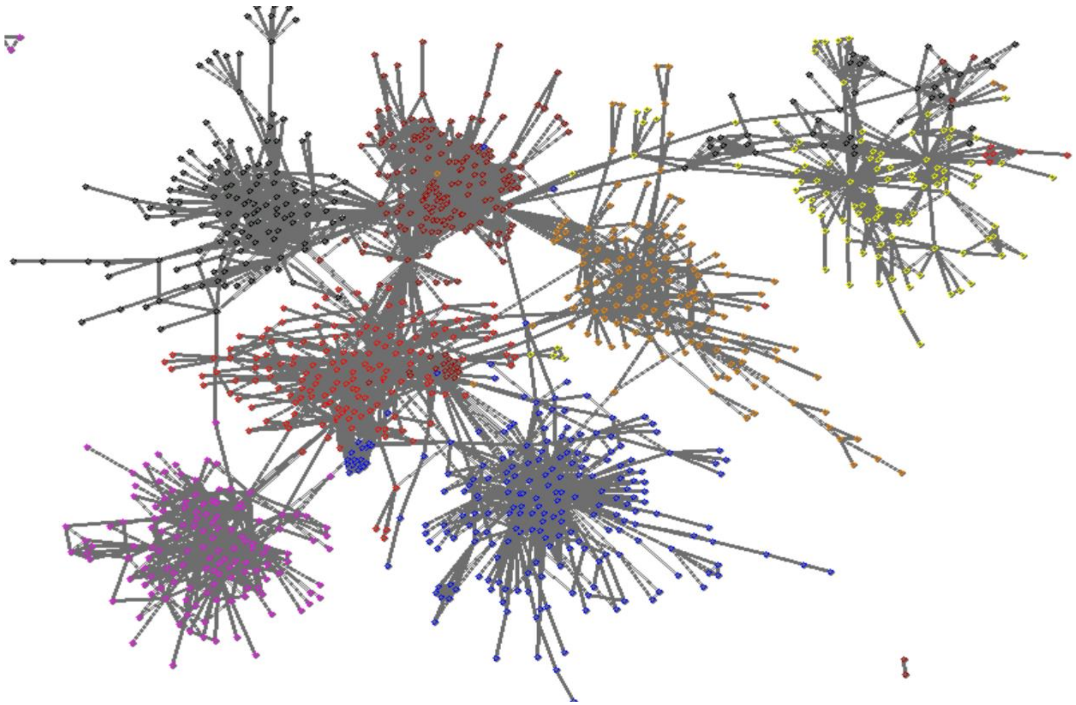
- Training data are given
- Error *E* depends on training data and network parameters (weights)
- Error *E* minimized by adapting weights so the model fits the training data
- Global minimum desirable

Chinese Whispers – Language Identification

- Data: words. Mixture of 7 languages; 100+ sentences each.
- Words have strong links if they frequently cooccur in the same sentence.

Co-occurrence graph:

- Each node is a word
- color corresponds to languages (after clustering)



Chris Biemann and Sven Teresniak. Disentangling from Babylonian Confusion – Unsupervised Language Identification, 2005

Chinese Whispers – Language Identification

- Algorithm results in 7 large clusters of words corresponding to each of the 7 languages.

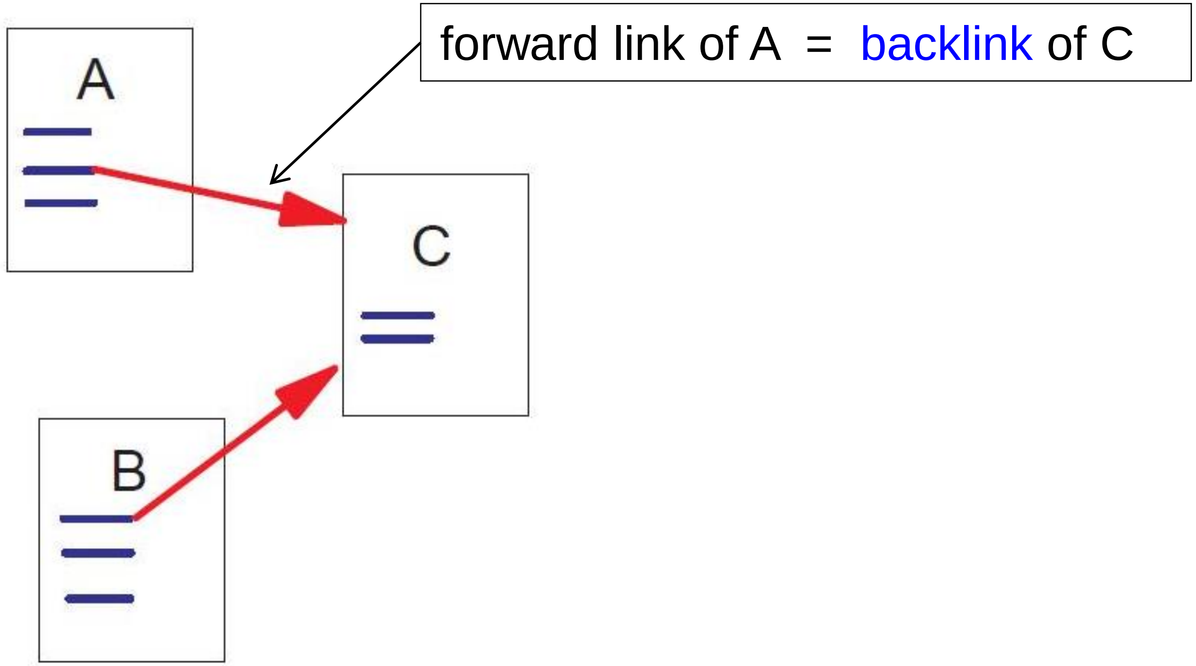
  - Dutch
  - English
  - Estonian
  - French
  - German
  - Icelandic
  - Italian

Overview

- Clustering Graphs
- SCAN
- Label Propagation
- Directed Graph: Webgraph Google

Mining the Web: Google (1)

- Webgraph: web page = vertex, weblink = edge
- Directed links, which can be seen as binary
- A web page is important if many pages refer to it *(~ vote)*



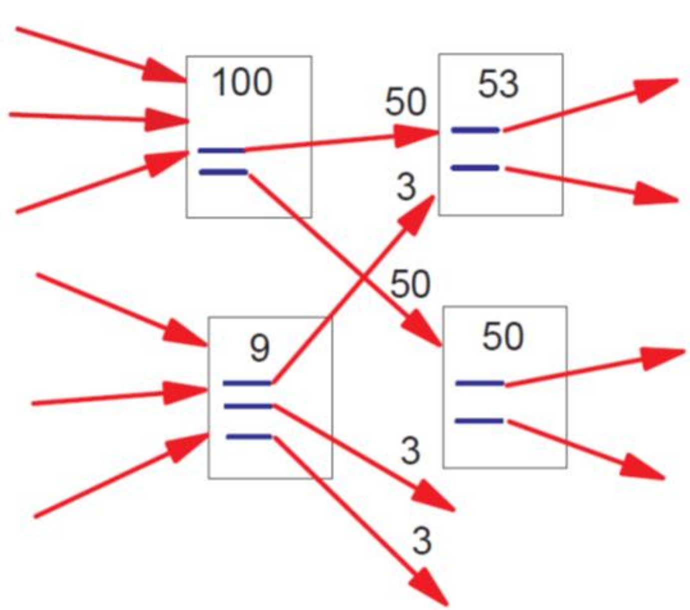
Brin, Page, et al. (1998) The PageRank Citation Ranking:
Bringing Order to the Web. Tech Rep Stanford Uni

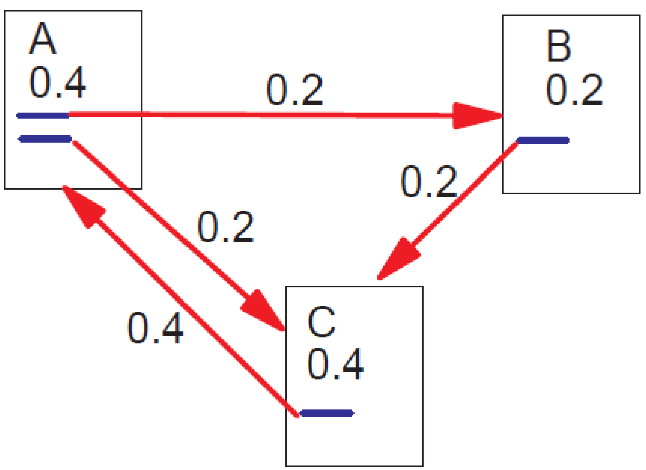
Google (2)

Ranking Function for web page *u*:

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}$$

v: web page that links to u
B_u: backlinks
$N_v = |F_v|$: # forward links from v
c: normalization factor



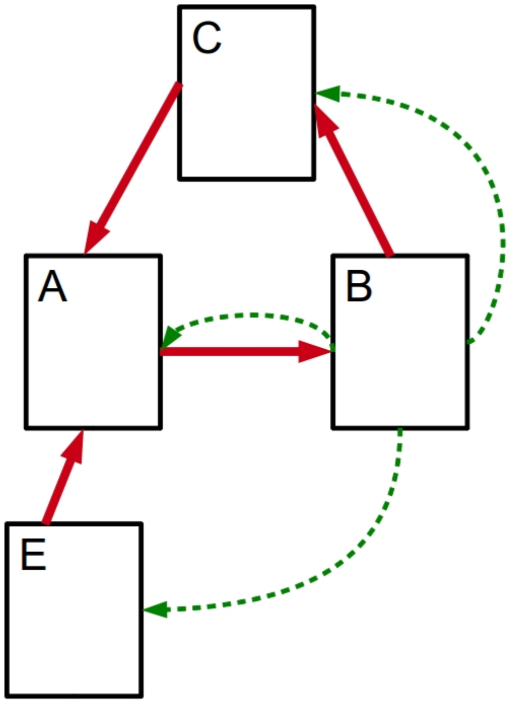


PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one

Google (3)

Problem: ***Rank Sink***
Some pages form a loop that accumulates rank to the infinity.

Solution: ***Random Surfer***
Jump to a random page based on some distribution *E*

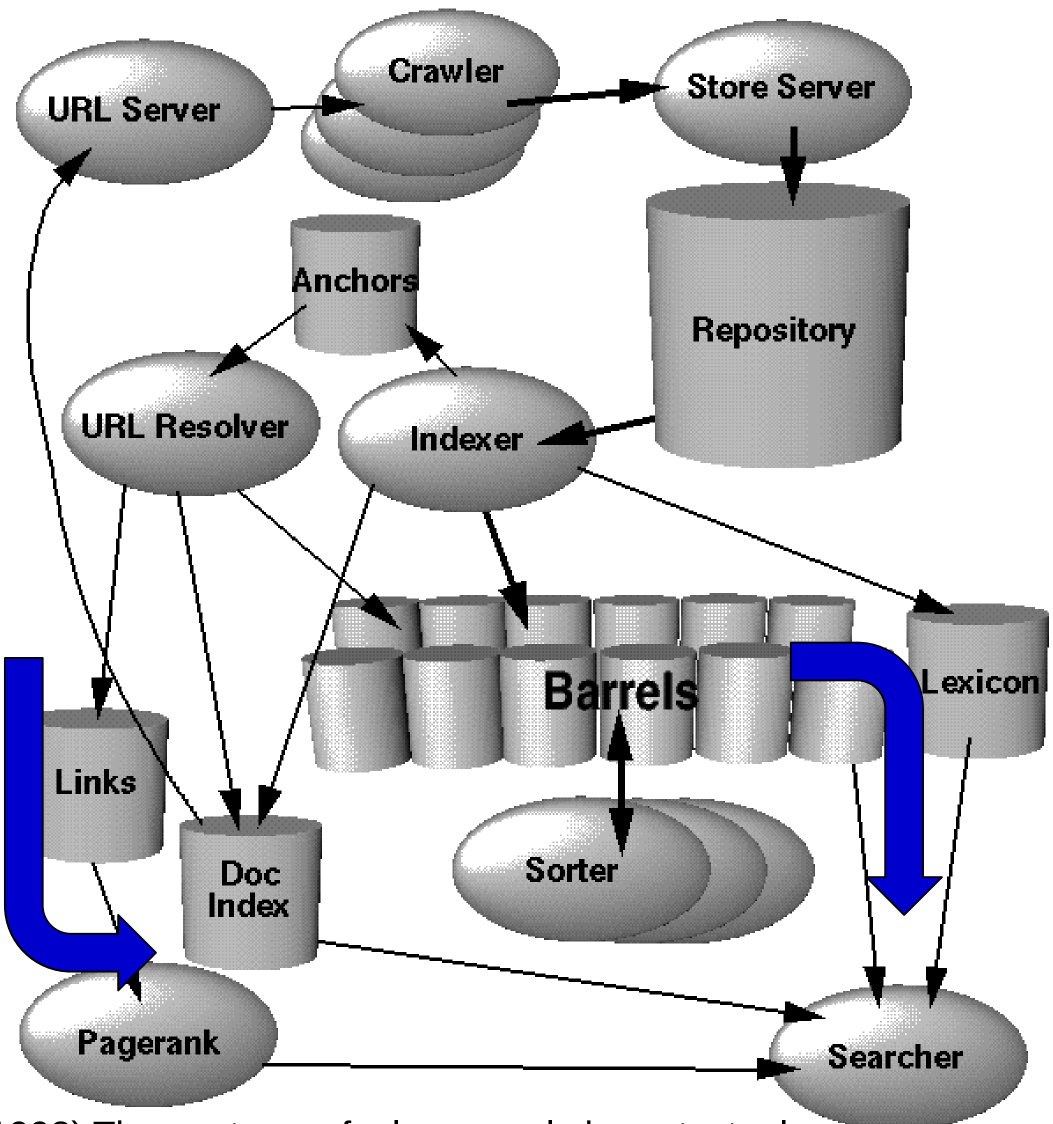


$$R'(u) = c \sum_{v \in B_u} \frac{R'(v)}{N_v} + cE(u)$$

rank source

Google (4)

ranks



word occurrences (hits)

Brin, Page (1998) The anatomy of a large-scale hypertextual Web search engine. Comp Netw and ISDN Syst

Summary

- Clustering graphs
  - sparsest cut – often intractable to find
  - SCAN – connect points iteratively based on structural similarity
  - Label propagation – Chinese Whispers algorithm without any parameters
- Google – its algorithm is transparent