Lecture 21

Graph Clustering Analysis

Slides adapted from multiple Internet sources:, intro to graph@NCSU, Monojit Choudhury@microsoft, Purnamitra Sarkar, Yang Zhou et.al, vldb2009

Outline

- Introduction to Clustering
- Types of Graph Clustering Analysis
- Algorithms for Graph Clustering
  - ❑ k-Spanning Tree
  - ❑ Shared Nearest Neighbor
  - ❑ Betweenness Centrality Based
  - ❑ Highly Connected Components
  - ❑ Maximal Clique Enumeration
  - ❑ Random Walk based Graph Clustering
- Application
  - – Overlapping Community Detection

2

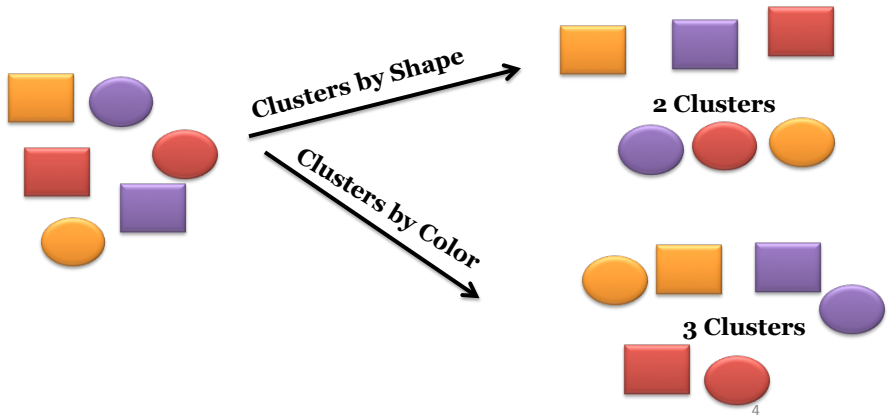
Outline

- Introduction to Clustering
- Introduction to Graph Clustering
- Algorithms for Graph Clustering
  - ❑ k-Spanning Tree
  - ❑ Shared Nearest Neighbor
  - ❑ Betweenness Centrality Based
  - ❑ Highly Connected Components
  - ❑ Maximal Clique Enumeration
  - ❑ Random Walk based Graph Clustering
- Application

3

What is Cluster Analysis?

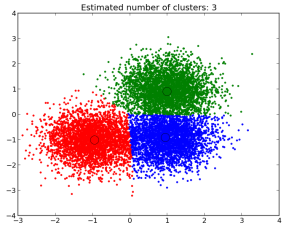
The process of dividing a set of input data into possibly overlapping, subsets, where elements in each subset are considered related by some similarity measure

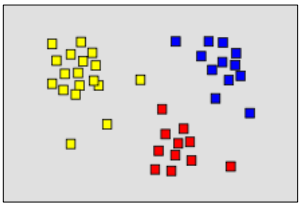


4

Clustering

- Group similar objects together
- Algorithms
  - K-Means, Fuzzy K-Means, Density-Based,…
- Different distance measures
  - Manhattan, Euclidean, …
- Summarization
  - Provides a macro-level view of the data-set





5

Outline

- Introduction to Clustering
- Introduction to Graph Clustering
- Algorithms for Graph Clustering
  - ❑ k-Spanning Tree
  - ❑ Shared Nearest Neighbor
  - ❑ Betweenness Centrality Based
  - ❑ Highly Connected Components
  - ❑ Maximal Clique Enumeration
  - ❑ Random Walk based Graph Clustering
- Application

6

What is Graph Clustering?

- Types
  - Between-graph
    - Clustering a set of graphs
  - Within-graph
    - Clustering the nodes/edges of a single graph

7

Between-graph Clustering

Between-graph clustering methods divide a set of graphs into different clusters

E.g., A set of graphs representing chemical compounds can be grouped into clusters based on their structural similarity

8

Within-graph Clustering

Within-graph clustering methods divides the nodes of a graph into clusters

E.g., In a social networking graph, these clusters could represent people with same/similar hobbies

In this lecture we will look at different algorithms to perform within-graph clustering

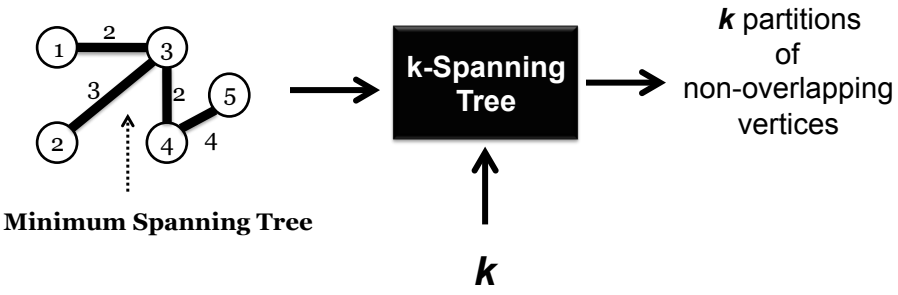
9

Outline

- Introduction to Clustering
- Introduction to Graph Clustering
- Algorithms for Within Graph Clustering
  - ❑ k-Spanning Tree
  - ❑ Shared Nearest Neighbor
  - ❑ Betweenness Centrality Based
  - ❑ Highly Connected Components
  - ❑ Maximal Clique Enumeration
  - ❑ Random Walk based Graph Clustering
- Application

10

k-Spanning Tree



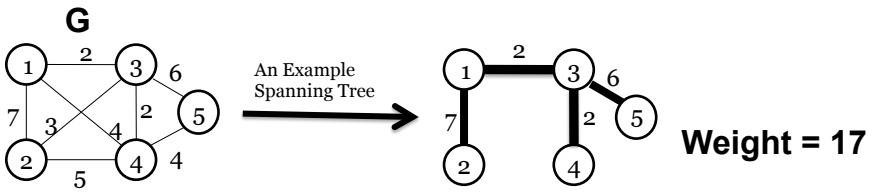
TWO STEPS:

- **Obtains the Minimum Spanning Tree (MST) of input graph G**
- **Removes k-1 edges from the MST and results in k clusters**

11

What is a Spanning Tree?

A connected subgraph with no cycles that includes all vertices in the graph

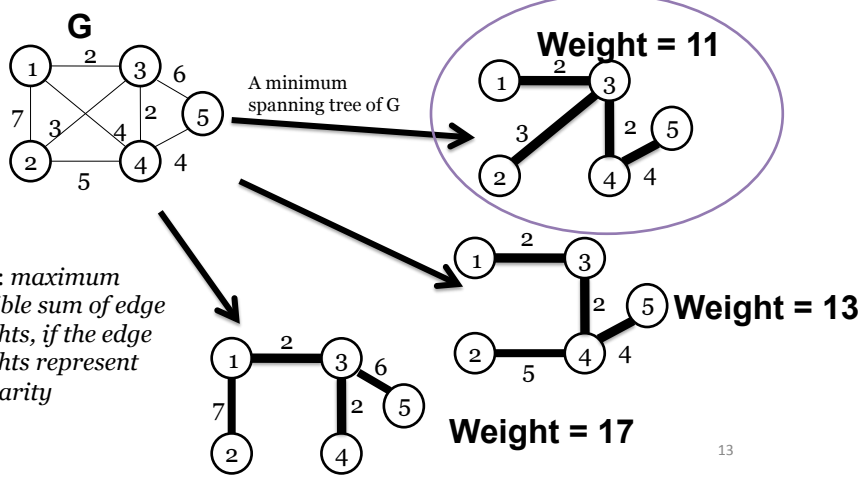


Note: *Weight can represent either distance between two vertices or similarity of the two vertices*

12

What is a Minimum Spanning Tree (MST)?

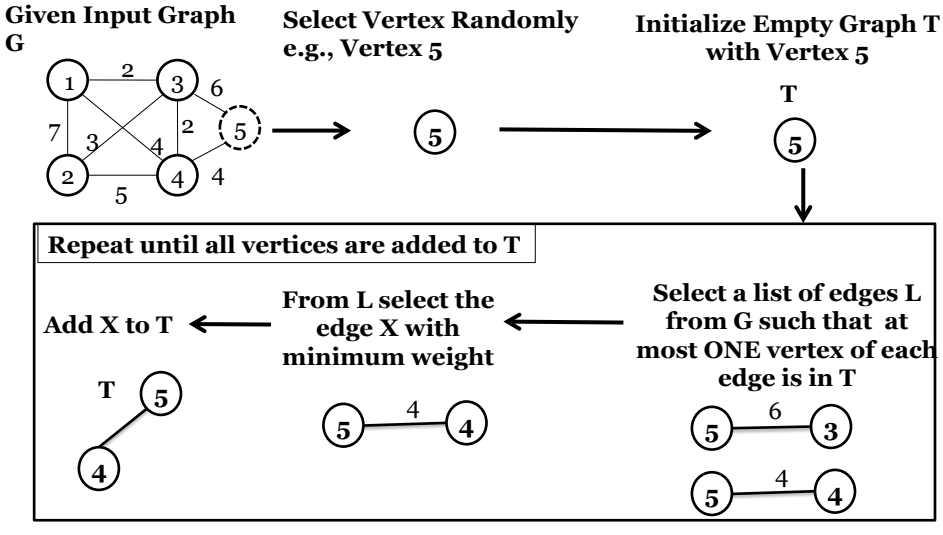
The spanning tree of a graph with the minimum possible sum of edge weights, if the edge weights represent distance



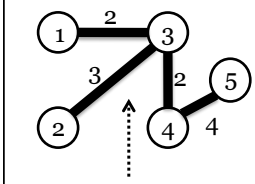
Note: *maximum possible sum of edge weights, if the edge weights represent similarity*

13

Algorithm to Obtain MST
Prim's Algorithm



k-Spanning Tree

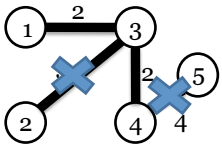


Minimum Spanning Tree

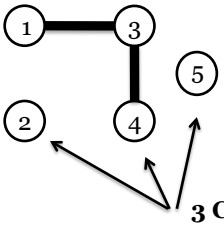
Remove k-1 edges with highest weight

Note: *k is the number of clusters*

E.g., k=3



E.g., k=3



3 Clusters

15

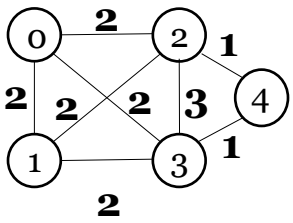
Outline

- Introduction to Clustering
- Introduction to Graph Clustering
- Algorithms for Within Graph Clustering
  - ❑ k-Spanning Tree
  - ❑ Shared Nearest Neighbor Clustering
  - ❑ Betweenness Centrality Based
  - ❑ Highly Connected Components
  - ❑ Maximal Clique Enumeration
  - ❑ Random Walk based Graph Clustering
- Application

17

Shared Nearest Neighbor Clustering

Shared Nearest Neighbor Graph (SNN)



Shared Nearest Neighbor Clustering

Groups of non-overlapping vertices

τ

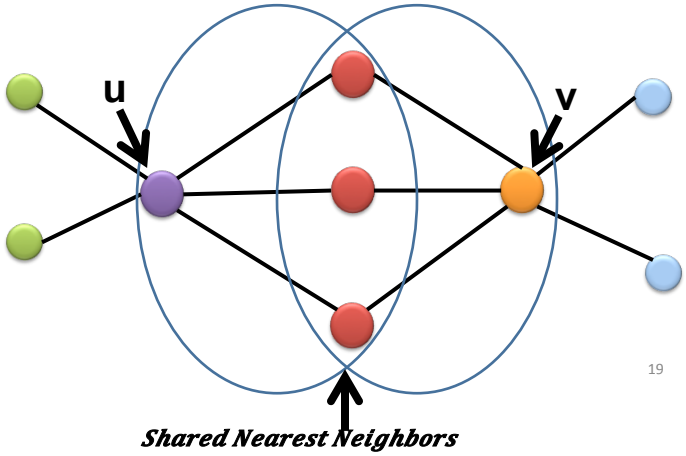
STEPS:

- **Obtains the Shared Nearest Neighbor Graph (SNN) of input graph G**
- **Removes edges from the SNN with weight less than τ**

18

What is Shared Nearest Neighbor? (Refresher from Proximity Chapter)

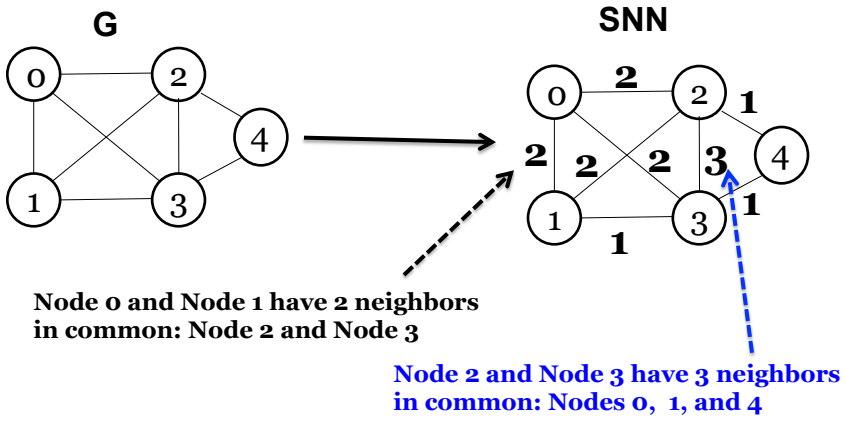
Shared Nearest Neighbor is a proximity measure and denotes the number of neighbor nodes common between any given pair of nodes



19

Shared Nearest Neighbor (SNN) Graph

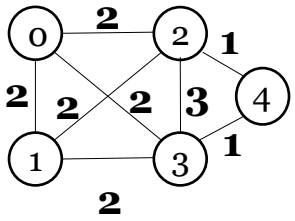
Given input graph G, weight each edge (u,v) with the number of shared nearest neighbors between u and v



20

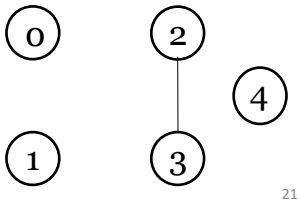
Shared Nearest Neighbor Clustering *Jarvis-Patrick Algorithm*

SNN graph of input graph G



If u and v share more than τ neighbors Place them in the same cluster

E.g., τ =3



Step 1:
- Obtains the Shared Nearest Neighbor Graph (SNN) of input graph G

Step 2
- Removes edges from the SNN with weight less than τ

21

Outline

- Introduction to Clustering
- Introduction to Graph Clustering
- Algorithms for Within Graph Clustering
  - ❑ k-Spanning Tree
  - ❑ Shared Nearest Neighbor Clustering
  - ❑ Betweenness Centrality Based
  - ❑ Highly Connected Components
  - ❑ Maximal Clique Enumeration
  - ❑ Random Walk based Graph Clustering
- Application

23

What is Betweenness Centrality?

Betweenness centrality quantifies the degree to which a vertex (or edge) occurs on the shortest path between all the other pairs of nodes

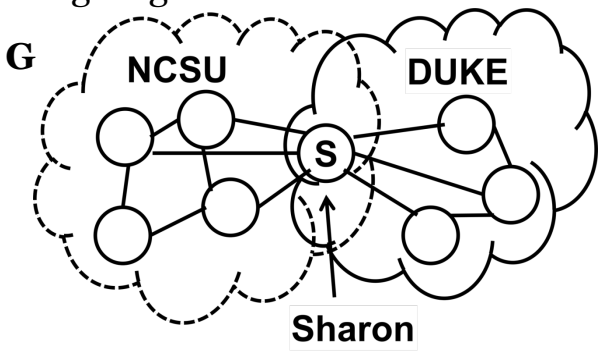
Two types:

- **Vertex Betweenness**
- **Edge Betweenness**

24

Vertex Betweenness

The number of **shortest paths in the graph G** that pass through a given node **S**

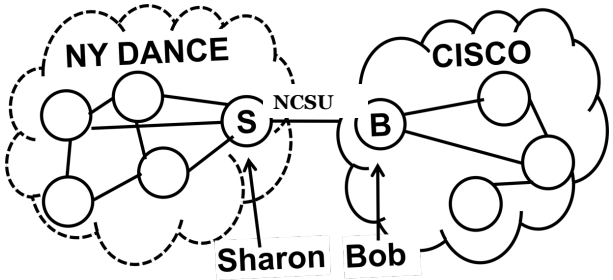


E.g., Sharon is likely a liaison between NCSU and DUKE and hence many connections between DUKE and NCSU pass through Sharon

25

Edge Betweenness

The number of **shortest paths in the graph G** that pass through given edge (S, B)



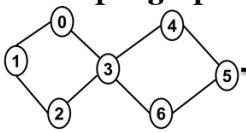
E.g., Sharon and Bob both study at NCSU and they are the only link between NY DANCE and CISCO groups

Vertices and Edges with high Betweenness form good starting points to identify clusters

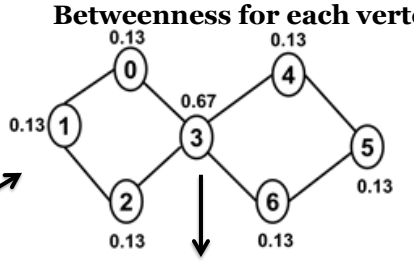
26

Vertex Betweenness Clustering

Given Input graph G

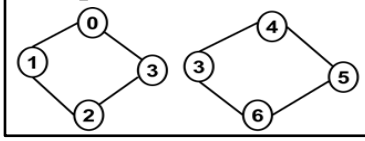


Betweenness for each vertex



Repeat until highest vertex betweenness ≤ μ

1. Disconnect graph at selected vertex (e.g., vertex 3)
2. Copy vertex to both Components

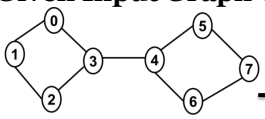


Select vertex v with the highest betweenness
E.g., Vertex 3 with value 0.67

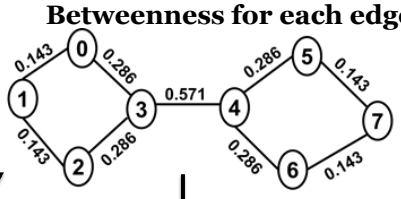
27

Edge-Betweenness Clustering
Girvan and Newman Algorithm

Given Input Graph G

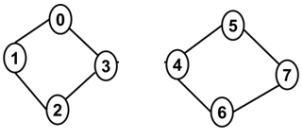


Betweenness for each edge



Repeat until highest edge betweenness ≤ μ

Disconnect graph at selected edge (E.g., (3,4))



Select edge with Highest Betweenness
E.g., edge (3,4) with value 0.571

29

Outline

- Introduction to Clustering
- Introduction to Graph Clustering
- Algorithms for Within Graph Clustering
  - ❑ k-Spanning Tree
  - ❑ Shared Nearest Neighbor Clustering
  - ❑ Betweenness Centrality Based
  - ❑ Highly Connected Components
  - ❑ Maximal Clique Enumeration
  - ❑ Random Walk based Graph Clustering
- Application

31

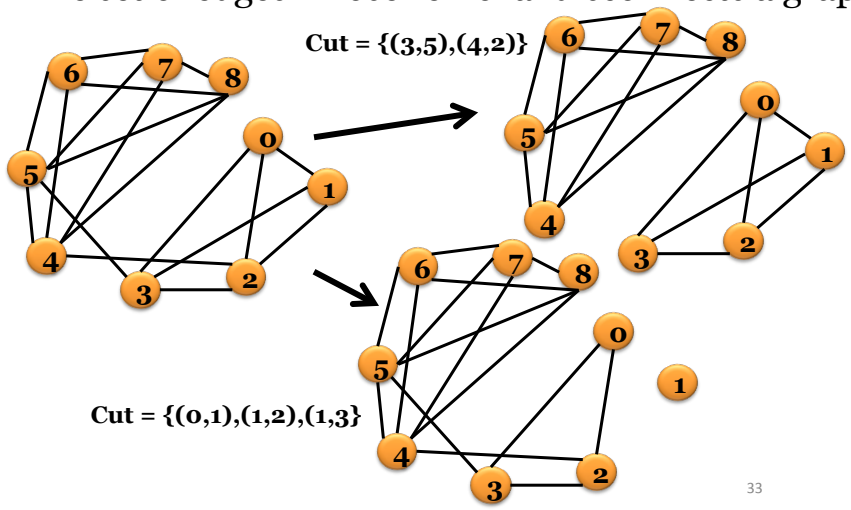
What is a Highly Connected Subgraph?

- Requires the following definitions
  - (Edge) Cut
  - Minimum Edge Cut (MinCut)
  - Edge Connectivity (EC)

32

Cut

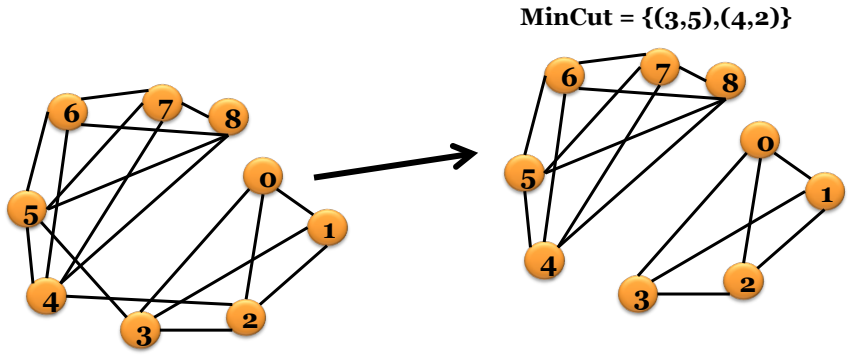
- The set of edges whose removal disconnects a graph



33

Minimum Cut

The minimum set of edges whose removal disconnects a graph

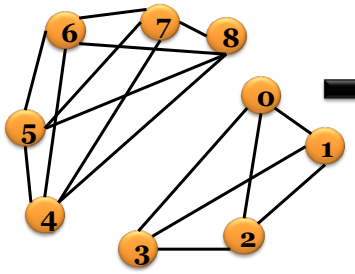


34

Edge Connectivity (EC)

- Minimum **NUMBER** of edges that will disconnect a graph

MinCut = {(3,5),(4,2)}



Edge Connectivity

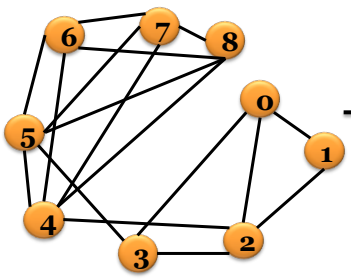
EC = | MinCut|
= | {(3,5),(4,2)}|
= 2

35

Highly Connected Subgraph (HCS)

A graph G =(V,E) is highly connected if EC(G)>V/2

G

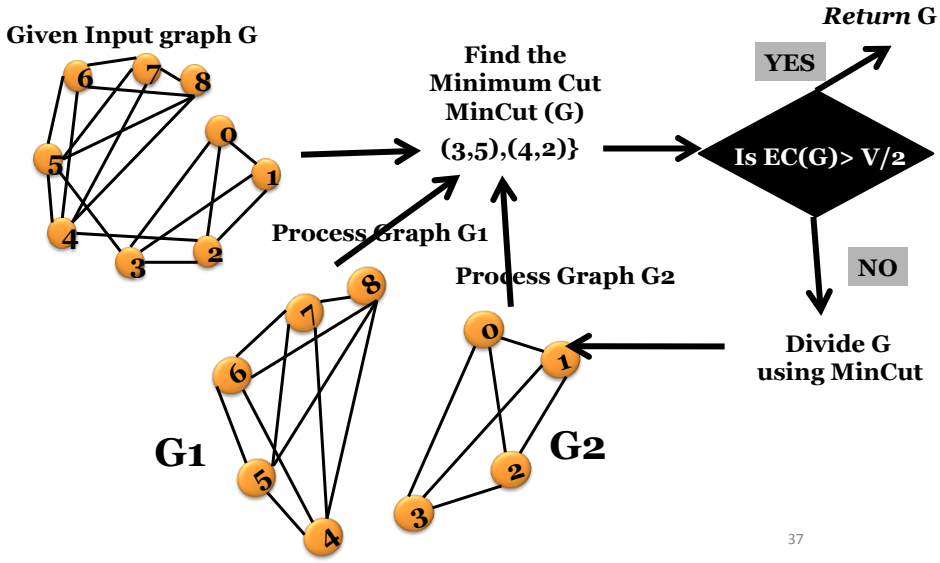


$EC(G) \not> V/2$
$2 \not> 9/2$

G is NOT a highly connected subgraph

36

HCS Clustering



37

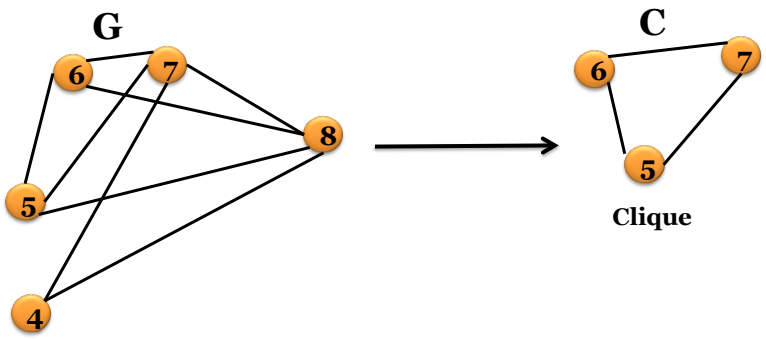
Outline

- Introduction to Clustering
- Introduction to Graph Clustering
- Algorithms for Within Graph Clustering
  - ❑ k-Spanning Tree
  - ❑ Shared Nearest Neighbor Clustering
  - ❑ Betweenness Centrality Based
  - ❑ Highly Connected Components
  - ❑ Maximal Clique Enumeration
  - ❑ Kernel k-means
- Application

39

What is a Clique?

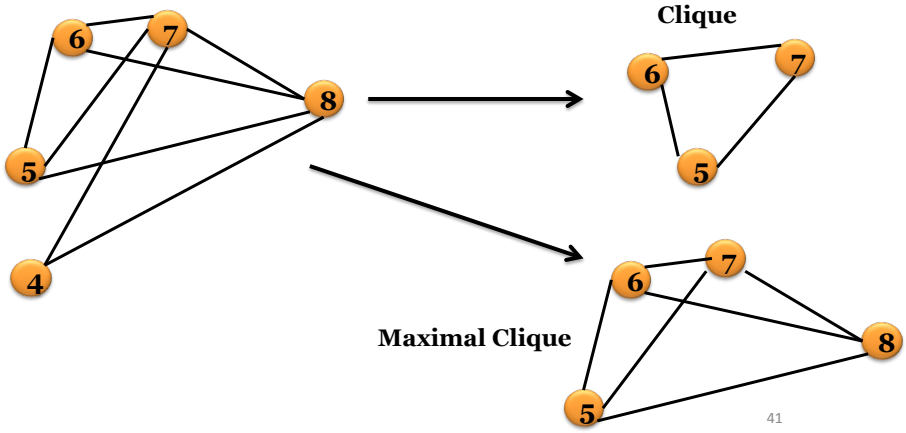
A subgraph C of graph G with edges between all pairs of nodes



40

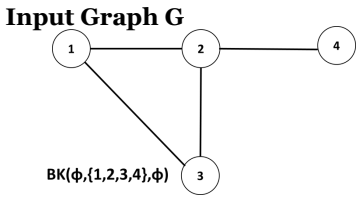
What is a Maximal Clique?

A maximal clique is a clique that is not part of a larger clique.



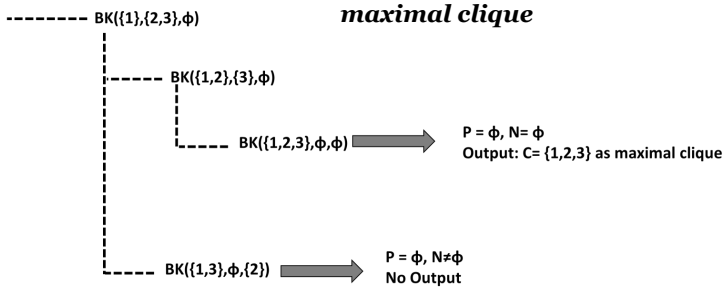
41

Maximal Clique Enumeration: *Bron and Kerbosch Algorithm*



BK(C,P,N)
C - vertices in current clique
P – vertices that can be added to C
N – vertices that cannot be added to C

Condition:
If both P and N are empty – output C as maximal clique



42

Outline

- Introduction to Clustering
- Introduction to Graph Clustering
- Algorithms for Within Graph Clustering
  - ❑ k-Spanning Tree
  - ❑ Shared Nearest Neighbor Clustering
  - ❑ Betweenness Centrality Based
  - ❑ Highly Connected Components
  - ❑ Maximal Clique Enumeration
  - ❑ Random Walk based Graph Clustering
- Application

44

What is a Random Walk

- Given a graph and a starting point (node), we select a neighbor of it at random, and move to this neighbor;
- Then we select a neighbor of this node and move to it, and so on;
- The (random) sequence of nodes selected this way is a random walk on the graph
- The number of hops in a random walk represents the random walk distance

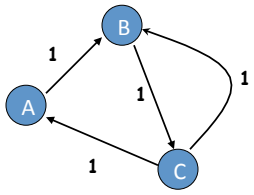
An example

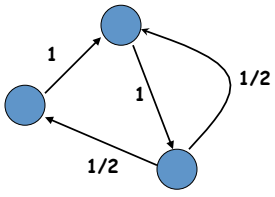
0	1	0
0	0	1
1	1	0

Adjacency matrix A

0	1	0
0	0	1
1/2	1/2	0

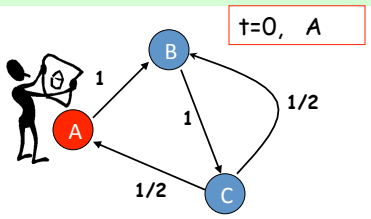
Transition matrix P





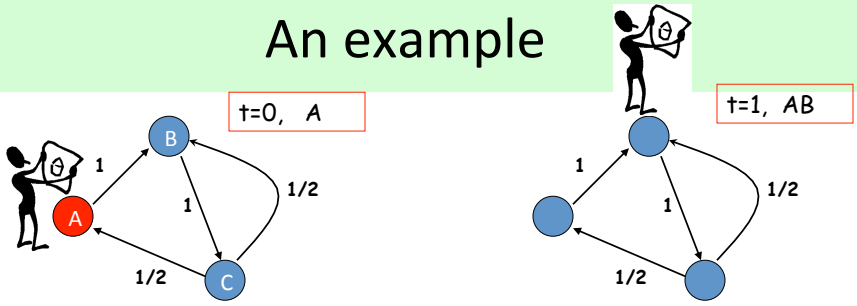
Slide from Purnamitra Sarkar, *Random Walks on Graphs: An Overview*

An example



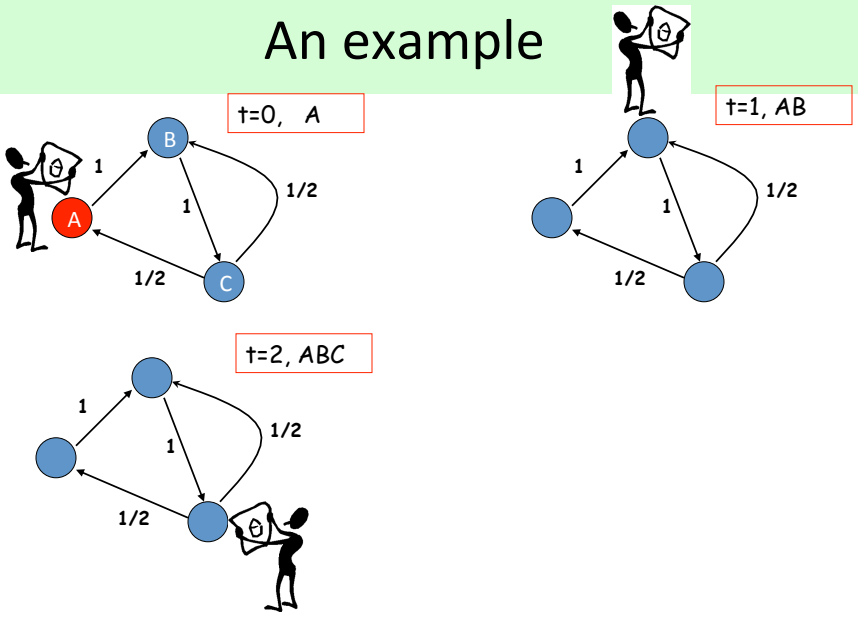
Slide from Purnamitra Sarkar, *Random Walks on Graphs: An Overview* 47

An example



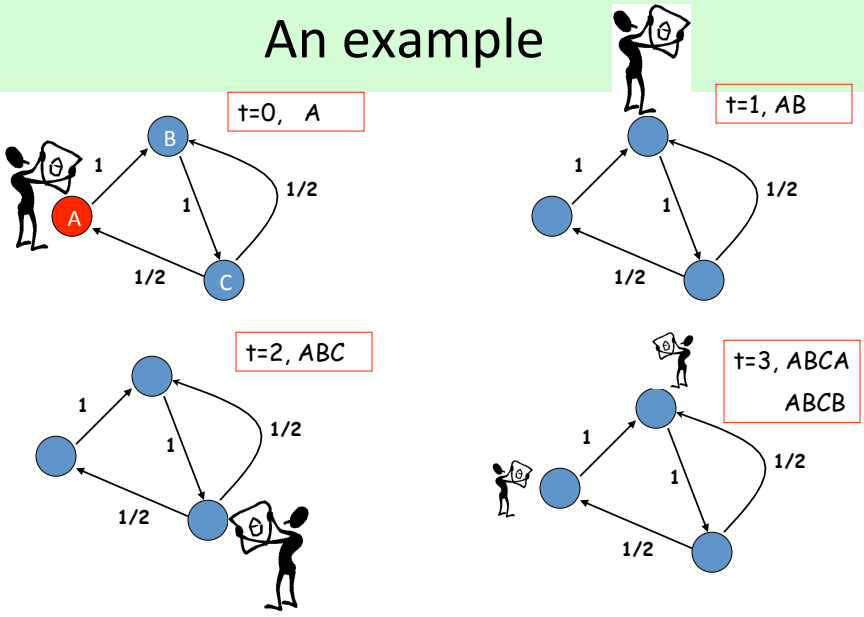
Slide from Purnamitra Sarkar, *Random Walks on Graphs: An Overview* 48

An example



Slide from Purnamitra Sarkar, *Random Walks on Graphs: An Overview*

An example



Slide from Purnamitra Sarkar, *Random Walks on Graphs: An Overview*

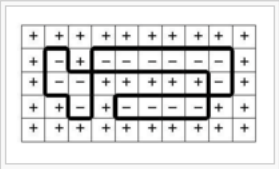
Why are random walks interesting?

- When the underlying data has a natural graph structure, several physical processes can be conceived as a random walk

Data	Process
WWW	Random surfer
Internet	Routing
P2P	Search
Social network	Information percolation

More examples

- Classic ones
  - Electrical circuits (resistances)
  - Brownian motion (random motion of particles)
  - Lattices and Ising models
- Not so obvious ones
  - Shuffling and permutations
  - Music
  - Language



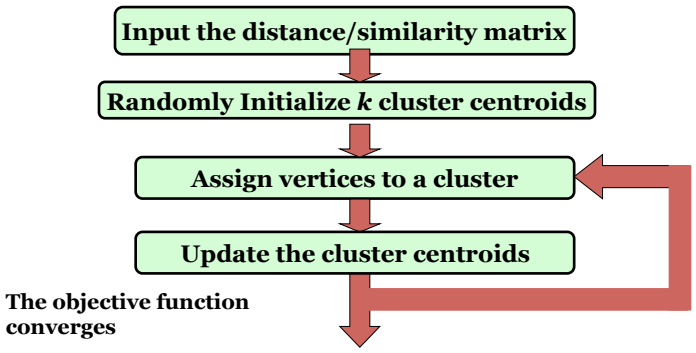
Spin configuration on a dual lattice

Graph Clustering: Random Walk based K-means

- Randomly select K vertices as the K initial centroids, one for each of the K clusters.
- For each vertex, compute the random walk distance (#hops) to each of the K cluster centroids and add vertex to the closest cluster by random walk distance, until all vertices are examined.
- Recompute the centroid for each of the K clusters
- Iterate the above process until the convergence condition is met.
  - ε% of vertices do not change their cluster home

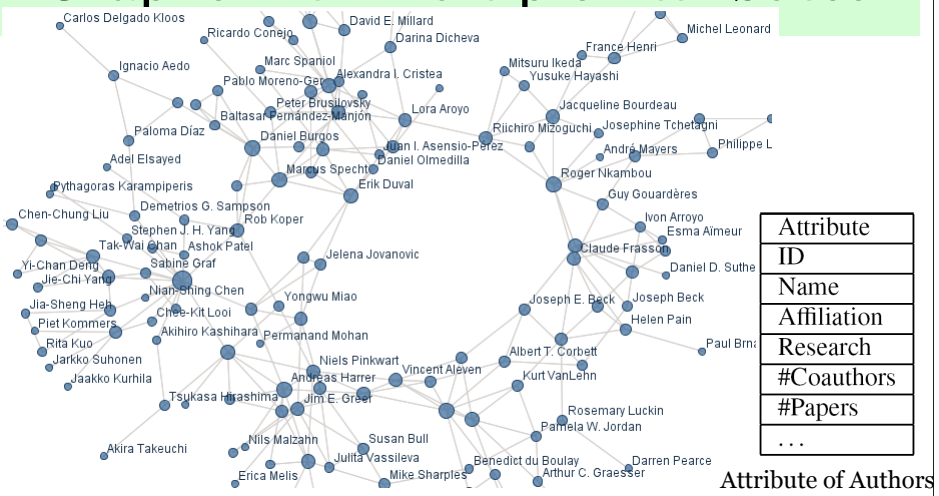
53

K-Medoids Clustering Framework



54

Graphs with Multiple Attributes



Coauthor Network of Top 200 Authors on TEL from DBLP
from manyeyes.alphaworks.ibm.com

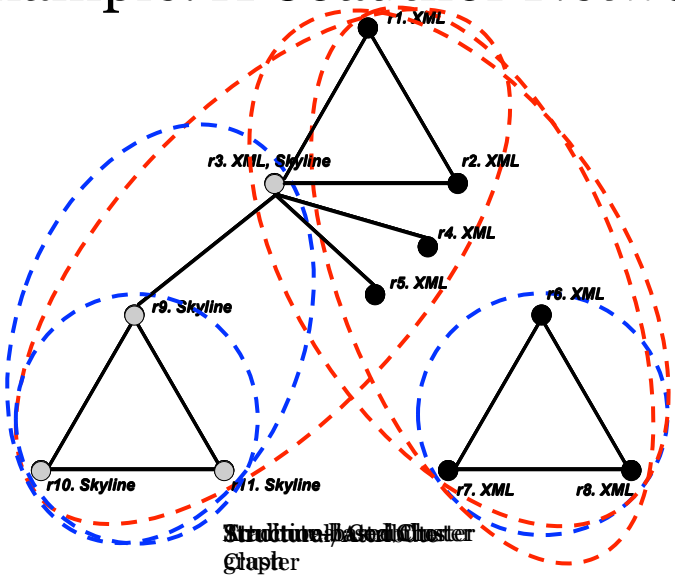
55

Graph Clustering Based on Structural and Attribute Similarities

- A desired clustering of attributed graph should achieve a good balance between the following:
  - **Structural cohesiveness**: Vertices within one cluster are close to each other in terms of structure, while vertices between clusters are distant from each other
  - **Attribute homogeneity**: Vertices within one cluster have similar attribute values, while vertices between clusters have quite different attribute values

56

Example: A Coauthor Network



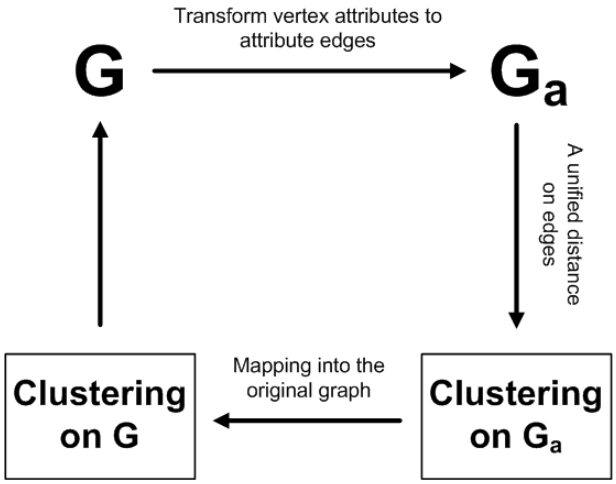
57

Different Clustering Approaches on the Graph with Multiple Attributes

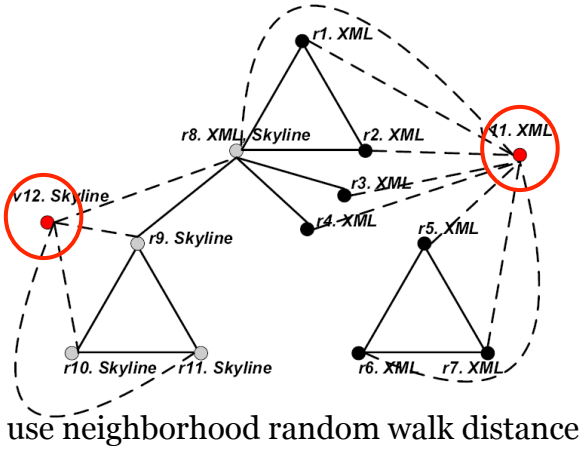
- Structure-based Clustering
  - Vertices with heterogeneous values in a cluster
- Attribute-based Clustering
  - Lose much structure information
- Structural/Attribute Cluster
  - Vertices with homogeneous values in a cluster
  - Keep most structure information

58

SA-Clustering Solution

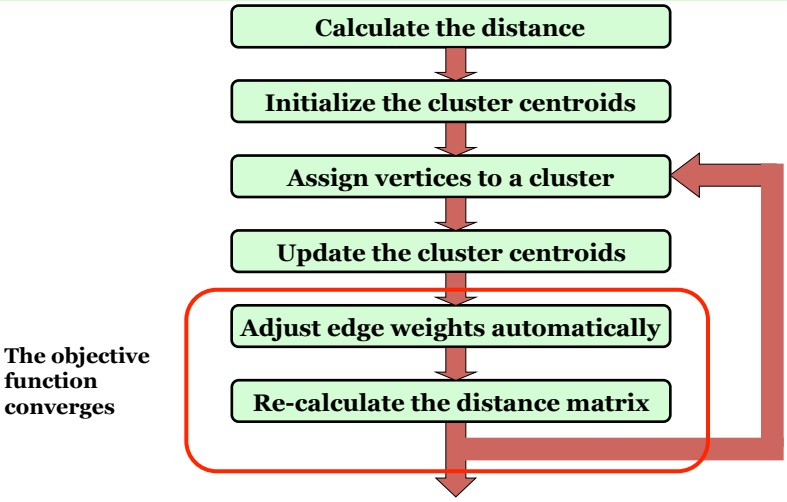


Attribute Augmented Coauthor Graph with Topics



Then we use neighborhood random walk distance on the augmented graph to combine structural and attribute similarities

New Clustering Framework



61

Distance Measure

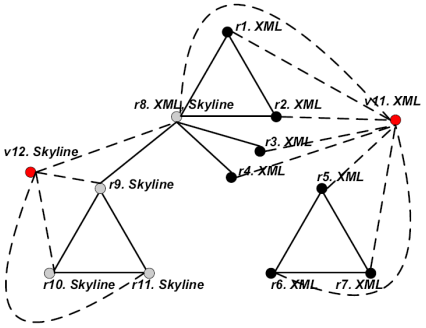
- Structural distance
  - Neighborhood random walk distance
- Attribute distance
  - e.g., Euclidean distance
- Hard to combine the two distances

$$d(v_i, v_j) = \alpha \cdot d_S(v_i, v_j) + \beta \cdot d_A(v_i, v_j)$$

62

Different Kinds of Vertices and Edges

- Two kinds of vertices
  - The Structure Vertex Set V
  - The Attribute Vertex Set V_a
- Two kinds of edges
  - The structure edges E
  - The attribute edges E_a



- The attribute augmented graph

$$G_a = (V \cup V_a, E \cup E_a)$$

Transition Probability Matrix on Attribute Augmented Graph

$$P_A = \begin{bmatrix} P_V & A \\ B & O \end{bmatrix}$$

P_V: probabilities from structure vertices to structure vertices

A: probabilities from structure vertices to attribute vertices

B: probabilities from attribute vertices to structure vertices

O: probabilities from attributes to attributes, all entries are zero

Clustering Process

- Assign each vertex $v_i \in V$ to its closest centroid c^* :

$$c^* = argmax_{c_j^t} d(v_i, c_j^t)$$

- Update the centroid with the most centrally located vertex in each cluster:
  - Compute the "average point" v_i of a cluster V_i

$$R_A^l(\overline{v_i}, v_j) = \frac{1}{|V_i|} \sum_{v_k \in V_i} R_A^l(v_k, v_j), \forall v_j \in V$$

  - Find the new centroid whose random walk distance vector is the closest to the cluster average

$$c_i^{t+1} = argmin_{v_j \in V_i} ||R_A^l(v_j) - R_A^l(\overline{v_i})||$$

67

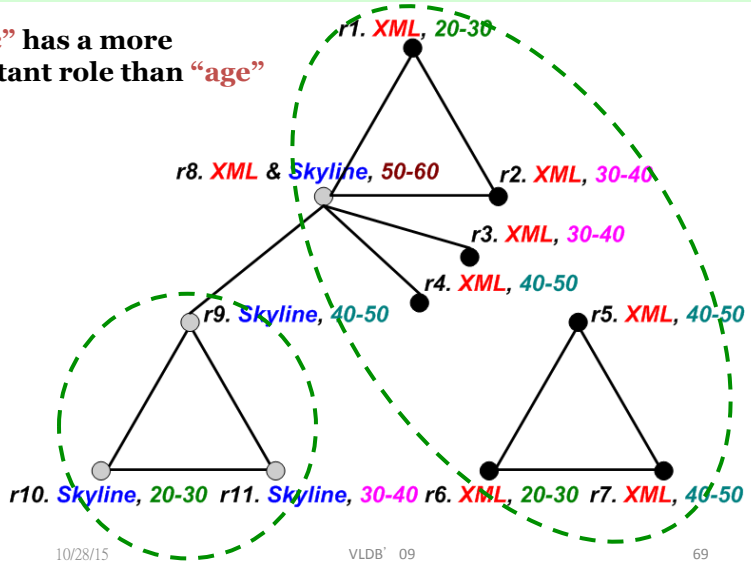
Edge Weight Definition

- Different types of edges may have different degrees of importance
  - Structure edge weight ω_0 fixed to 1.0 in the whole clustering process
  - Attribute edge weight ω_i for attribute a_i, $i = 1,2,\ldots,m$
    - Different attributes have different importance/contributions to the clustering objective
  - All weights are initialized to 1.0, but will be automatically updated during clustering

68

Clustering A Graph with Two Attributes

"Topic" has a more important role than "age"



Weight Self-Adjustment

- A vote mechanism determines whether two vertices share an attribute value:

$$vote_i(v_p, v_q) = \begin{cases} 1, & if\ v_p,\ v_q\ share\ the\ same\ value\ on\ a_i \\ 0, & otherwise \end{cases}$$

- Weight Increment:

$$\triangle\omega_i^t = \frac{\sum_{j=1}^{k} \sum_{v \in V_j} vote_i(c_j, v)}{\frac{1}{m} \sum_{p=1}^{m} \sum_{j=1}^{k} \sum_{v \in V_j} vote_p(c_j, v)}$$

- How the weight adjustment affects clustering convergence?

Clustering Convergence

- Graph Clustering Objective Function:

$$O(\{V_i\}_{i=1}^{k}, W) = \sum_{i=1}^{k} d(V_i, V_i)$$

- Interpretation

  Demonstrate that the weights are adjusted towards the direction of clustering convergence when we iteratively refine the clusters.

- Theorem

  Given a certain partition $\{V_i^*\}_{i=1}^{k}$ of graph G, there exists a unique solution $W^* = \{\omega_1^*, ..., \omega_m^*\}$ which maximizes the objective function.

71

Experimental Evaluation

- Datasets
  - Political Blogs Dataset: 1490 vertices, 19090 edges, one attribute *political leaning*
  - DBLP Dataset : 5000 vertices, 16010 edges, two attributes *prolific* and *topic*
- Methods
  - K-SNAP [Tian et al., SIGMOD'08]: attribute only
  - S-Cluster: structure-based clustering
  - W-Cluster: $d(v_i, v_j) = \alpha \cdot d_S(v_i, v_j) + \beta \cdot d_A(v_i, v_j)$
  - SA-Cluster: our proposed method

72

Evaluation Metrics

- **Density**: intra-cluster structural cohesiveness

$$density(\{V_i\}_{i=1}^{k}) = \sum_{i=1}^{k} \frac{|\{(v_p, v_q)|v_p, v_q \in V_i, (v_p, v_q) \in E\}|}{|E|}$$

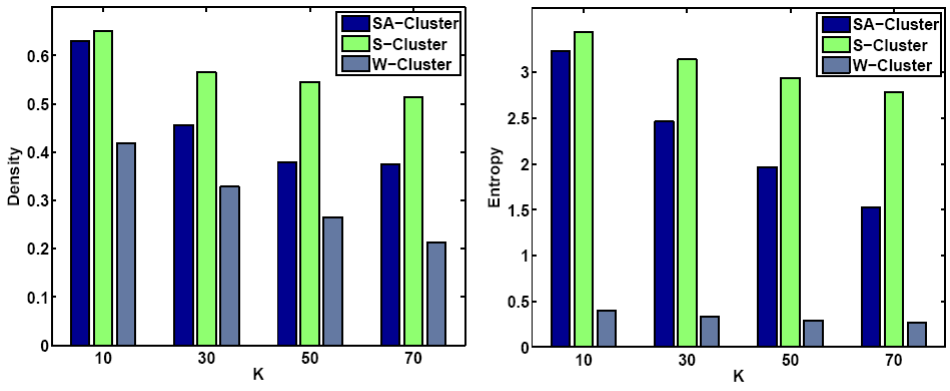
- **Entropy**: intra-cluster attribute homogeneity

$$entropy(\{V_i\}_{i=1}^{k}) = \sum_{i=1}^{m} \frac{\omega_i}{\sum_{p=1}^{m} \omega_p} \sum_{j=1}^{k} \frac{|V_j|}{|V|} entropy(a_i, V_j)$$

where

$$entropy(a_i, V_j) = -\sum_{n=1}^{n_i} p_{ijn} log_2 p_{ijn}$$

Cluster Quality Evaluation

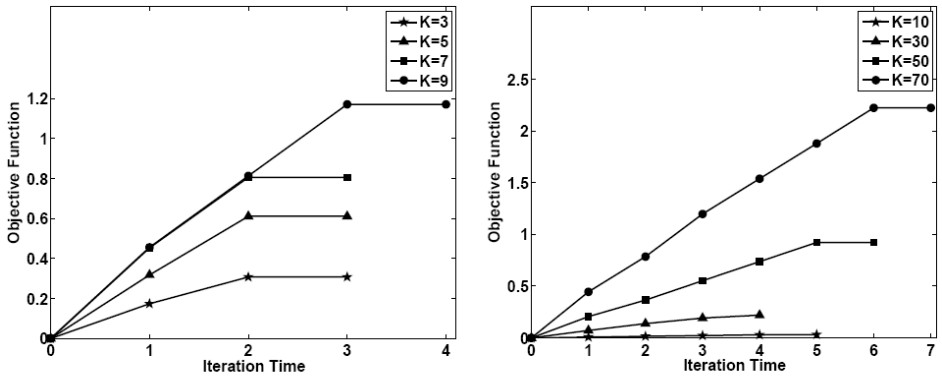


(a) density (b) entropy

Cluster Quality Comparison on DBLP

74

Clustering Convergence



(a) Political Blogs (b) DBLP

Cluster Convergence

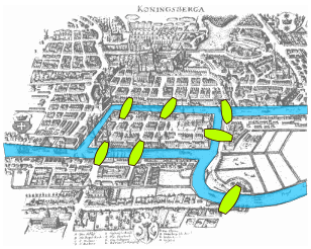
75

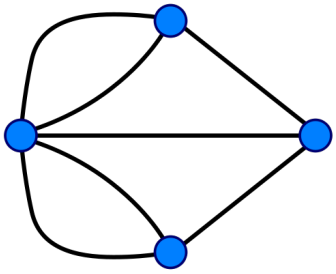
Outline

- Introduction to Clustering
- Introduction to Graph Clustering
- Algorithms for Within Graph Clustering
  - k-Spanning Tree
  - Shared Nearest Neighbor Clustering
  - Betweenness Centrality Based
  - Highly Connected Components
  - Maximal Clique Enumeration
  - Kernel k-means
- Application
  - Real Graphs
  - Overlapping Community Detection

76

Graphs from the Real World

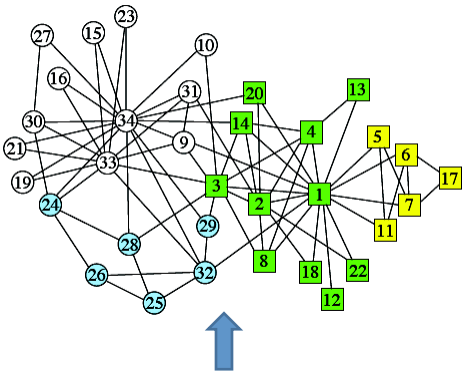




Königsberg's Bridges

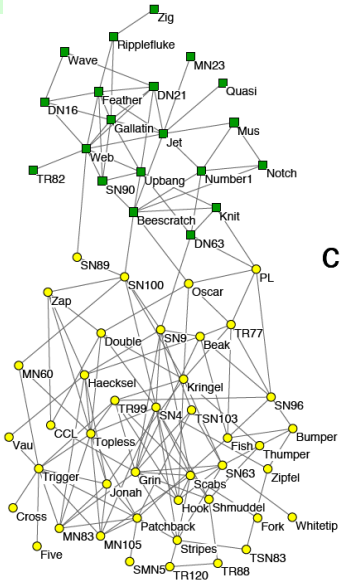
Ref: http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

Graphs from the Real World



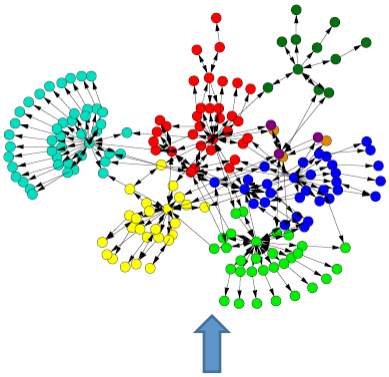
Zachary's Karate Club

Lusseau's network of bottlenose dolphins



C

Graphs from the Real Word

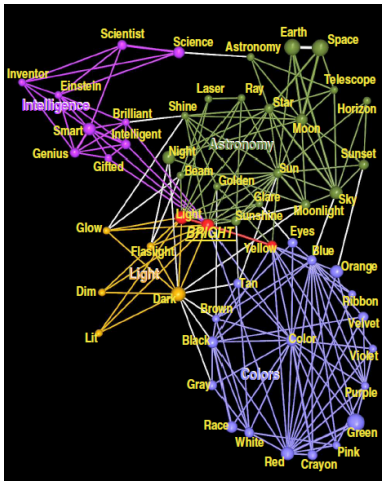


Webpage Hyperlink Graph

Directed Communities

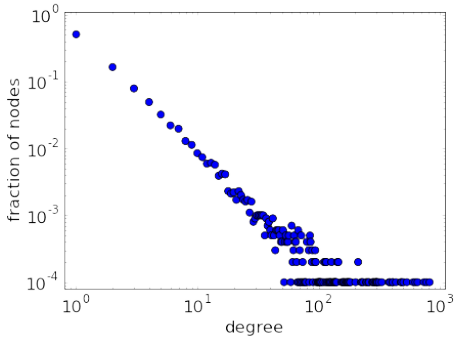
Network of Word Associations

Overlapping Communities



Real Networks Are Not Random

- Degree distribution is broad, and often has a tail following power-law distribution

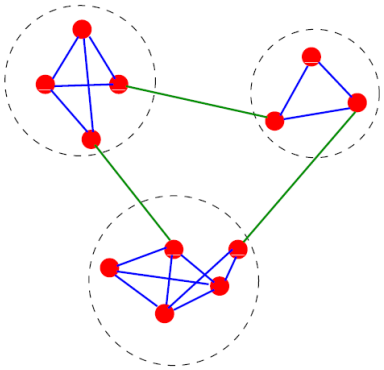


$$P(d) = cd\uparrow a$$
$$\Rightarrow \log(P(d)) = \log(c) + a\log(d)$$

Ref: "Plot of power-law degree distribution on log-log scale." From Math Insight. http://mathinsight.org/image/power_law_degree_distribution_scatter

Real Networks Are Not Random

- Edge distribution is locally inhomogeneous



Community Structure!

Applications of Graph Clustering Community Detection

- Website mirror server assignment
- Recommendation system
- Social network role detection
- Functional module in biological networks
- Graph coarsening and summarization
- Network hierarchy inference

General Challenges

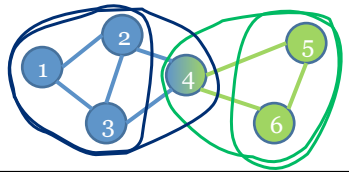
- Structural clusters can only be identified if graphs are sparse (i.e. $m=O(n)$)
  - Motivation for graph sampling/sparsification
- Many clustering problems are **NP**-hard. Even polynomial time approaches may be too expensive
  - Call for scalable solutions
- Concepts of "cluster", "community" are not quantitatively well defined
  - Discussed in more details below

Overlapping community detection

- Most of previous methods can only generate non-overlapped clusters.
  - A node only belongs to one community.
  - Not real in many scenarios.
    - A person usually belongs to multiple communities.
- Most of current overlapping community detection algorithms can be categorized into three groups.
  - Mainly based on non-overlapping communities algorithms.
  - Bridge nodes, line graphs, local clustering

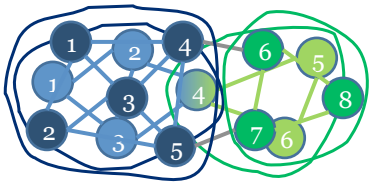
Overlapping community detection

- 1. Identifying bridge nodes
  - First, identifying bridge nodes and remove or duplicate these nodes.
    - Duplicate nodes have connection b/t them.
  - Then, apply hard clustering algorithm.
    - If bridge nodes was removed, add them back.
  - E.g. DECAFF [Li2007], Peacock [Gregory2009]
  - Cons: Only a small part of nodes can be identified as bridge nodes.



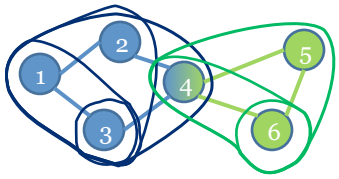
Overlapping community detection

- 2. Line graph transformation
  - Edges become nodes.
    - New nodes have connection if they originally share a node.
  - Then, apply hard clustering algorithm on the line graph.
  - E.g. LinkCommunity [Ahn2010]
  - Cons: An edge can only belong to one cluster



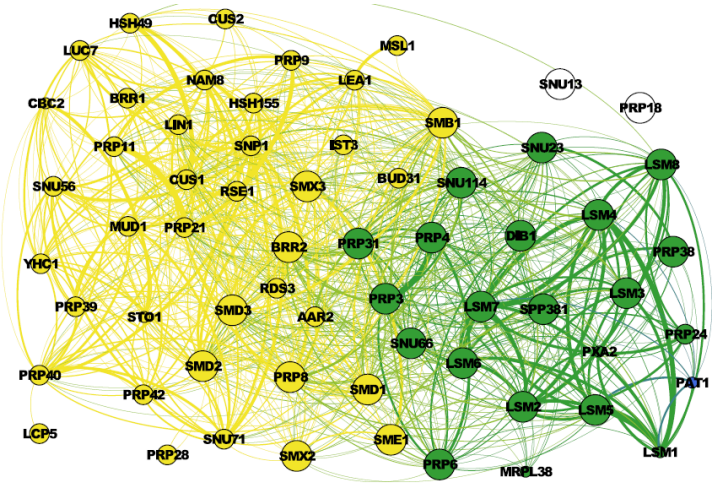
Overlapping community detection

- 3. Local clustering
  - (optional) Select seed nodes.
  - Expand seed node according to some criterion.
  - E.g. ClusterOne [Nepusz2012], MCODE [Bader2003], CPM [Adamcsek2006], RRW [Macropol2009]
  - Cons: Not globally consider the topology



Multi-resolution methods

- Many graphs have a hierarchical cluster structure.



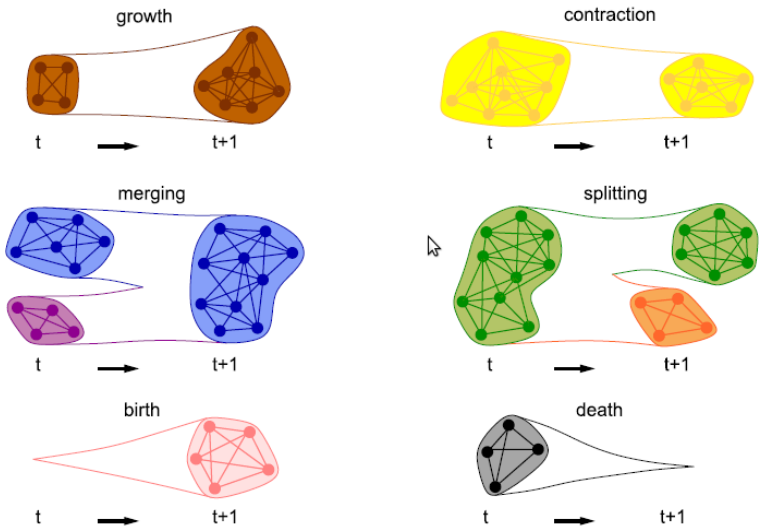
Multi-resolution / Hierarchical methods

- Most of previous methods can only generate a clustering with fixed resolution (avg. cluster size)
  - Clusters might be hierarchical or users might be interesting in different resolutions.
- Multi-resolution methods
  - Produce clusterings with different average cluster size.
- Hierarchical Clustering
  - Produce a dendrogram, showing the hierarchical clusters.

Dynamic community

- Cluster each snapshot independently
- Then mapping clusters in each clustering.
  - If two clusters in continuous snapshots share most of nodes, then the next one evolves from the previous one.
- Detect the evolution of communities in a dynamic graph.
  - Birth, Death, Growth, Contraction, Merge, Split.

Dynamic community



Dynamic community

- Asur et al. (2007) further detect an event involving nodes.
  - E.g. join and leave
  - **Measure the node behavior**.
    - Sociability: How frequently a node join and leave a community.
    - Influence: How a node can influence other nodes' activities.
- Usage
  - **Understand the community behavior.**
    - E.g. age is positively correlated with the size.
  - **Predict the evolution of a community**
  - Predict node (user) behavior, predict link

Dynamic community detection

- Hypothesis: Communities in dynamic graphs are "smooth".
  - Detect communities by also considering the previous snapshots.
- Chakrabarti et al (2006) introduce history cost.
  - Measures the dissimilarity between two clusterings in continuous timestamps.
  - A smooth clustering has lower history cost.
  - Add this cost to the objective function.

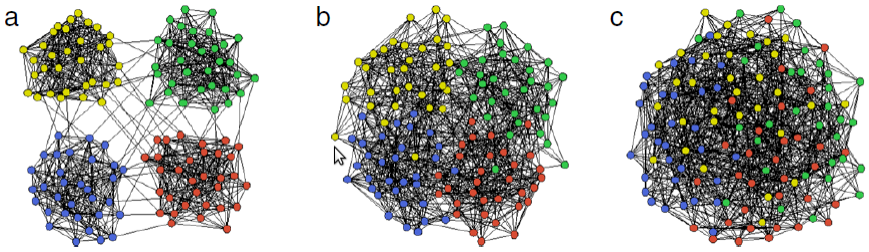
Testing algorithms

- 1. Real data w/o gold standards:
- 2. Read data w/ gold standard
- 3. Synthetic data

- Hard to say which algorithm is the best.
  - In different scenarios, different algorithms might be best choices.
- 1 and 2 are practical, but hard to determine which kinds of graphs / clusters an algorithm is suitable.
  - Sparse/Dense, power-law, overlapping communities.

Synthetic data

- Girvan and Newman (2002) Benchmark
  - Fixed 128 nodes and 4 communities
  - Can tune noisy level



  - Cons: All nodes have the same expected degree; All communities have the same size, etc

Synthetic data

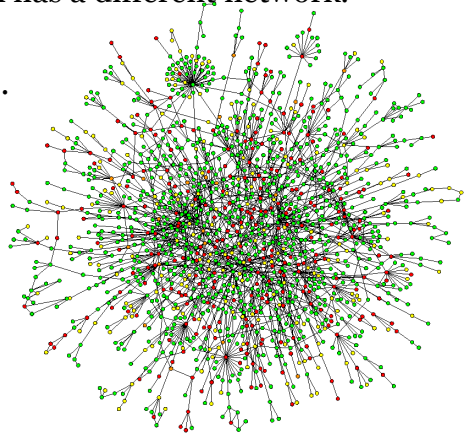
- LFR (Lancichinetti 2009)
  - Generate power-law, weighted/unweighted, directed/ undirected graph with gold standard
  - Pros: can generate variaous graphs.
    - # nodes, average degree, power-law exponent.
    - Average/Min/Max community size, # bridge nodes.
    - Noisy level, etc.
  - Cons: The number of communities each bridge node belonging to is fixed.
- Use the above metrics to evaluate the result.

Biological Application

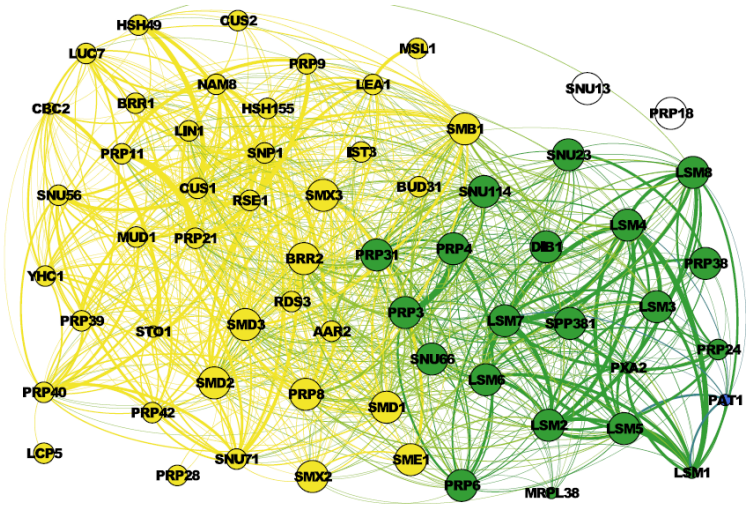
- Protein-protein interaction (PPI) network
  - Node: Protein; Edge: Interaction
  - Edge weight: Confidence level of an interaction
  - Interacting proteins are likely to have the same function.
  - Community: Protein complex or functional module
    - Gene Ontology terms, etc.
- Usage: Predict functions of each protein
  - Biologically examining each protein is expensive
  - Improve drug design, etc.

PPI networks

- Usually thousands of nodes.
  - Each dataset, organism has a different network.
  - average degree 5~10.
  - power-law distribution.

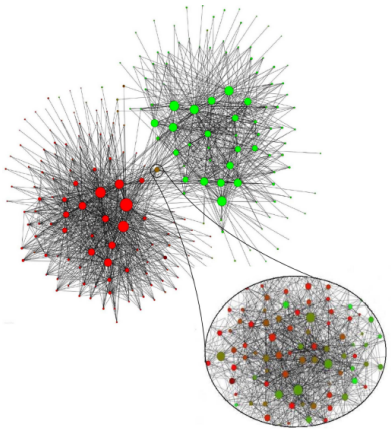


PPI sub-network example



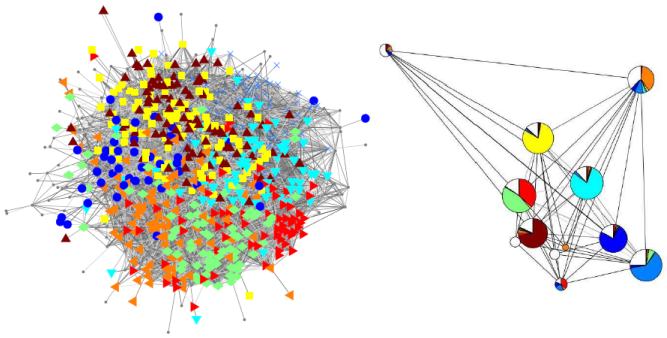
Social Network Applications

- Social Networks
  - Belgian phone call network distinguishes French- and Dutch-speaking population



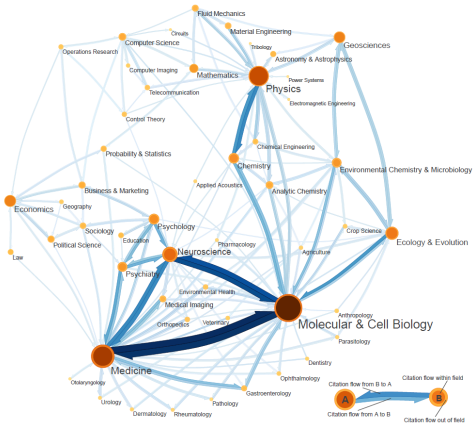
Social Network Applications

- Social Networks
  - University students Facebook network (left) and corresponding dorm affiliation (right)



Citation Network Applications

- Other Networks
  - "Map of science" derived from citation network



Questions?