Graph Mining - Motivation, Applications and Algorithms

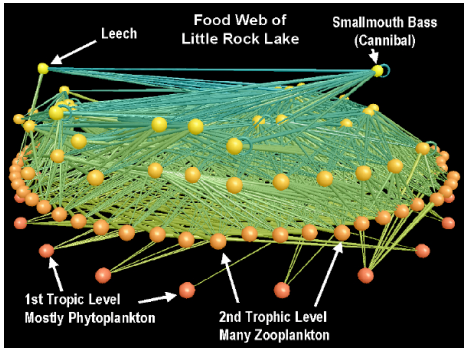
大数据分析 | 何铁科

http: // hetieke.ml

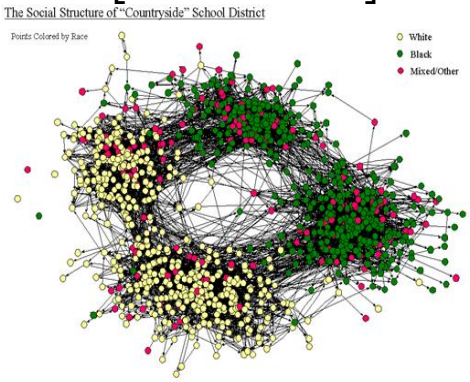
南京大学 NANJING UNIVERSITY

Graphs - why should we care?

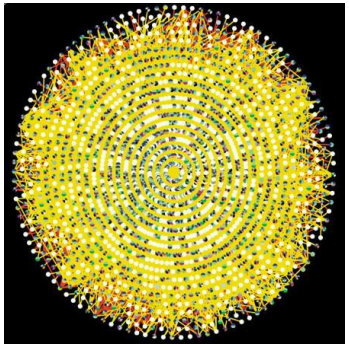
Internet Map
[lumeta.com]



Food Web
[Martinez '91]



Friendship Network
[Moody '01]



Protein Interactions
[genomebiology.com]

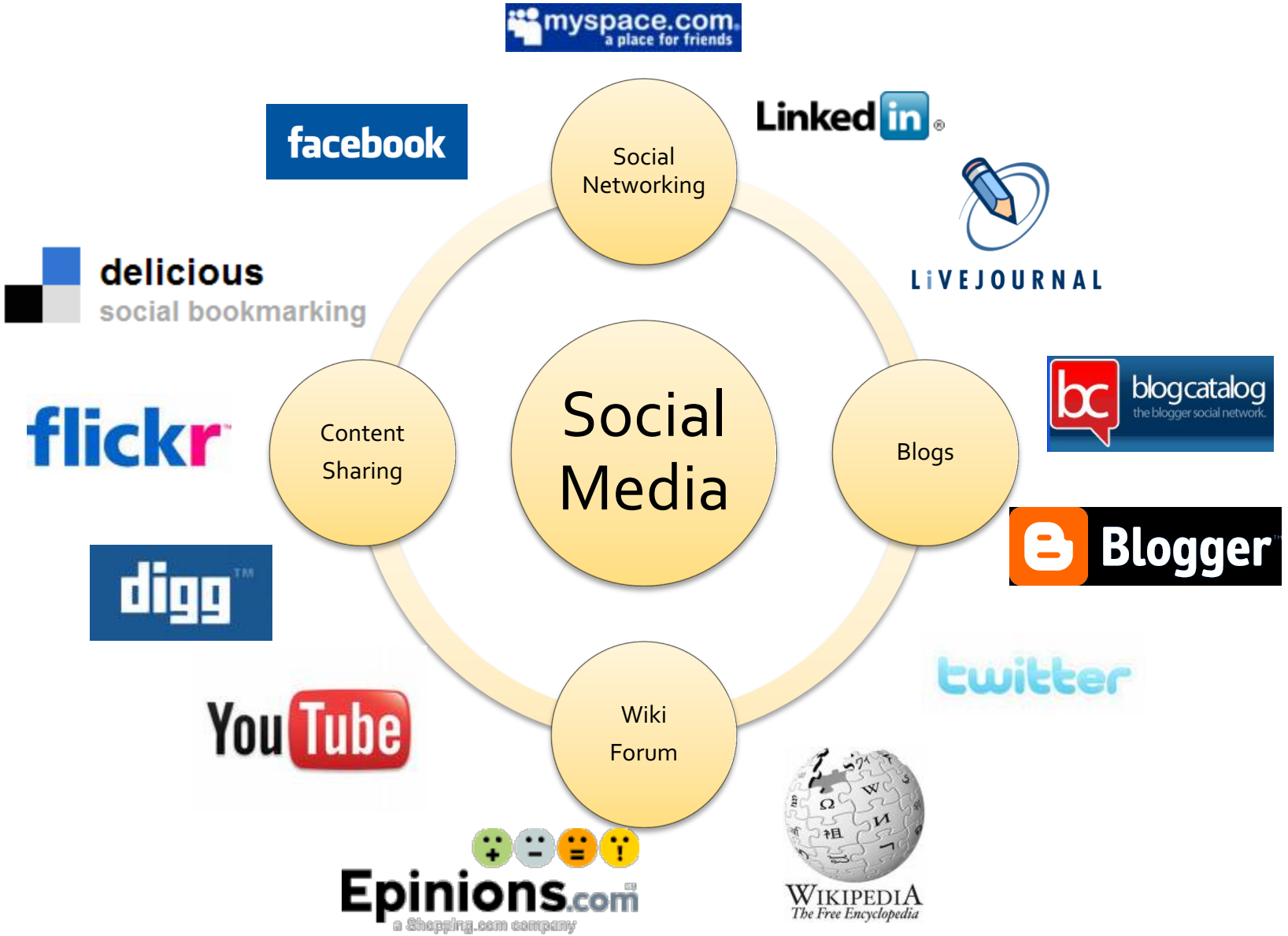
PARTICIPATING WEB AND SOCIAL MEDIA

Traditional Media

Broadcast Media: One-to-Many

Communication Media: One-to-One

Social Media: Many-to-Many



Characteristics of Social Media

- Everyone can be a media outlet
- Disappearing of communications barrier
  - Rich User Interaction
  - User-Generated Contents
  - User Enriched Contents
  - User developed widgets
  - Collaborative environment
  - Collective Wisdom
  - Long Tail

Broadcast Media
Filter, then Publish

Social Media
Publish, then Filter

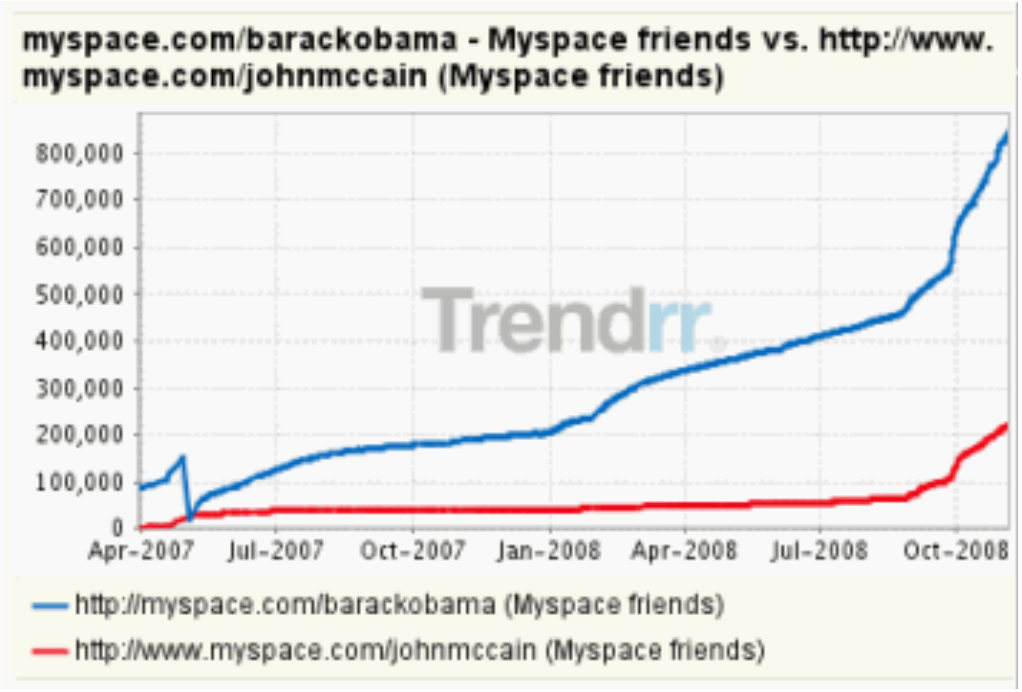
Top 20 Most Visited Websites

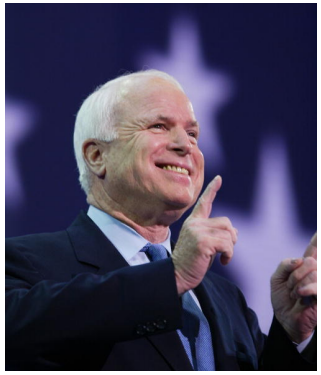
- Internet traffic report by Alexa on August 27th, 2009

1	Google	11	MySpace
2	Yahoo!	12	Google India
3	Facebook	13	Google Germany
4	YouTube	14	Twitter
5	Windows Live	15	QQ.Com
6	Wikipedia	16	RapidShare
7	Blogger	17	Microsoft Corporation
8	Microsoft Network (MSN)	18	Google France
9	Baidu.com	19	WordPress.com
10	Yahoo! Japan	20	Google UK

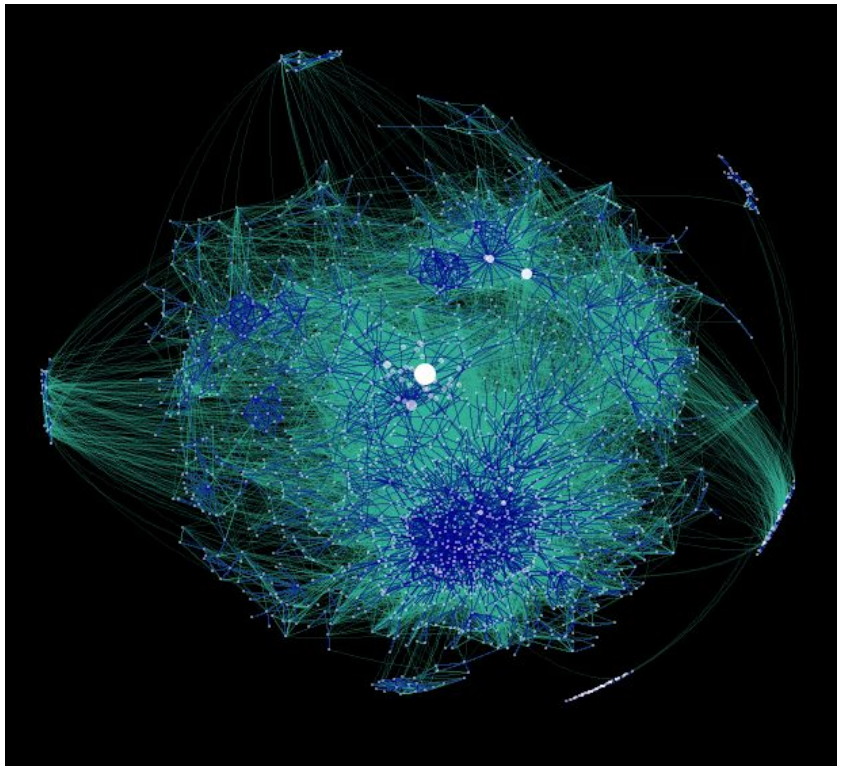
- 40% of the top 20 websites are social media sites

Social Media's Important Role



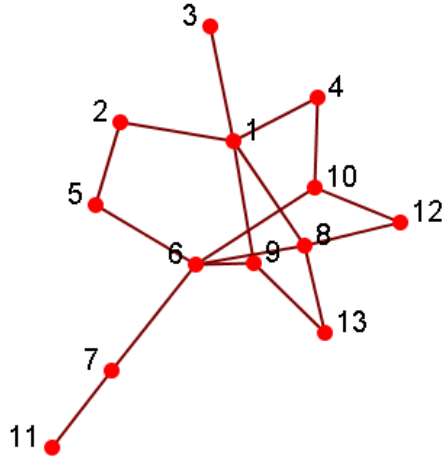


"social networks will complement, and may replace, some government functions,"



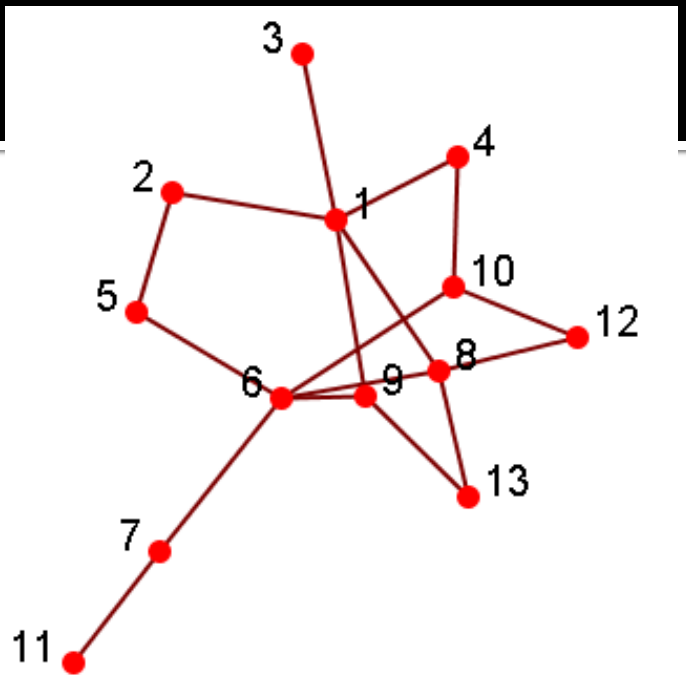
Social Networks

- A social structure made of nodes (individuals or organizations) that are related to each other by various interdependencies like friendship, kinship, etc.
- Graphical representation
  - Nodes = members
  - Edges = relationships
- Various realizations
  - Social bookmarking (Del.icio.us)
  - Friendship networks (facebook, myspace)
  - Blogosphere
  - Media Sharing (Flickr, Youtube)
  - Folksonomies



Sociomatrix

Social networks can also be represented in matrix form



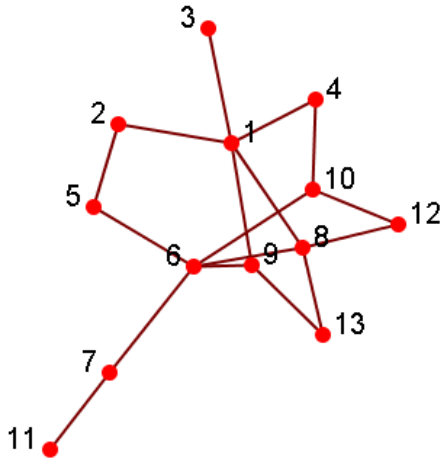
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	1	1	1	0	0	0	1	1	0	0	0	0
2	1	0	0	0	1	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0	0
…													

Social Computing and Data Mining

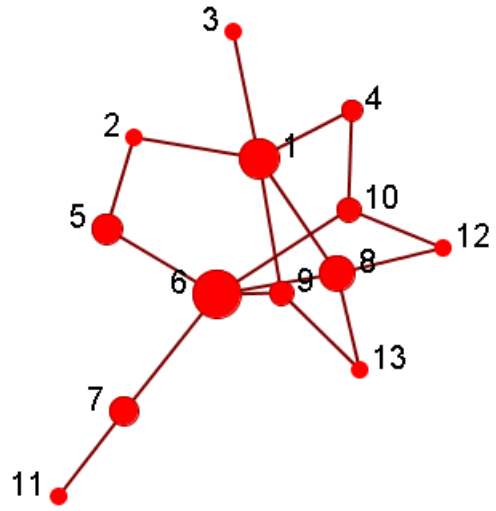
- Social computing is concerned with the study of social behavior and social context based on computational systems.
- Data Mining Related Tasks
  - Centrality Analysis
  - Community Detection
  - Classification
  - Link Prediction
  - Viral Marketing
  - Network Modeling

Centrality Analysis/Influence Study

- Identify the most important actors in a social network
- Given: a social network
- Output: a list of top-ranking nodes



Top 5 important nodes:
6, 1, 8, 5, 10



(Nodes resized by Importance)

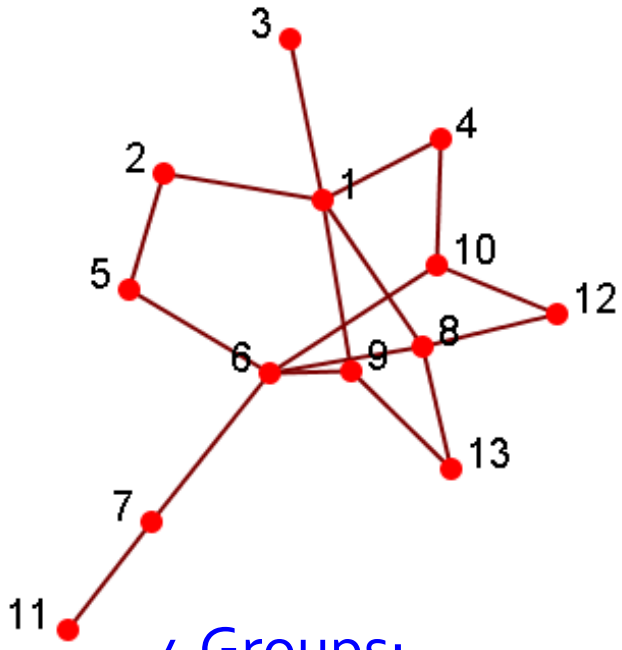
Community Detection

- A community is a set of nodes between which the interactions are (relatively) frequent

  a.k.a. group, subgroup, module, cluster
- Community detection

  a.k.a. grouping, clustering, finding cohesive subgroups
  - Given: a social network
  - Output: community membership of (some) actors
- Applications
  - Understanding the interactions between people
  - Visualizing and navigating huge networks
  - Forming the basis for other tasks such as data mining

Visualization after Grouping



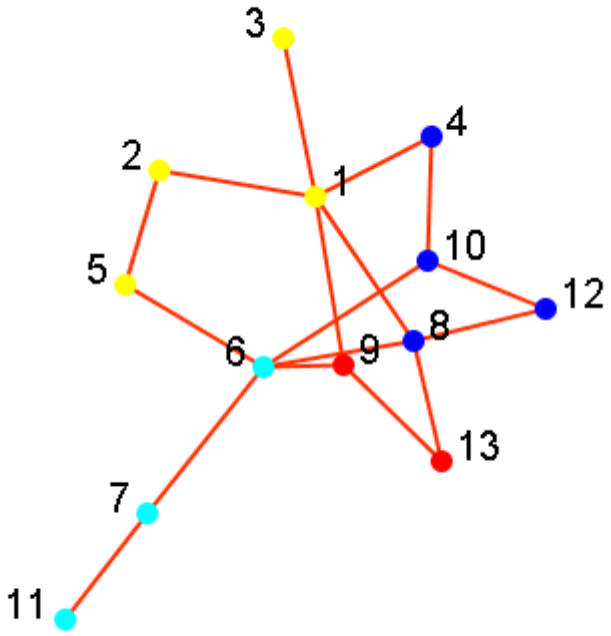
4 Groups:

{1,2,3,5}

{4,8,10,12}

{6,7,11}

{9,13}

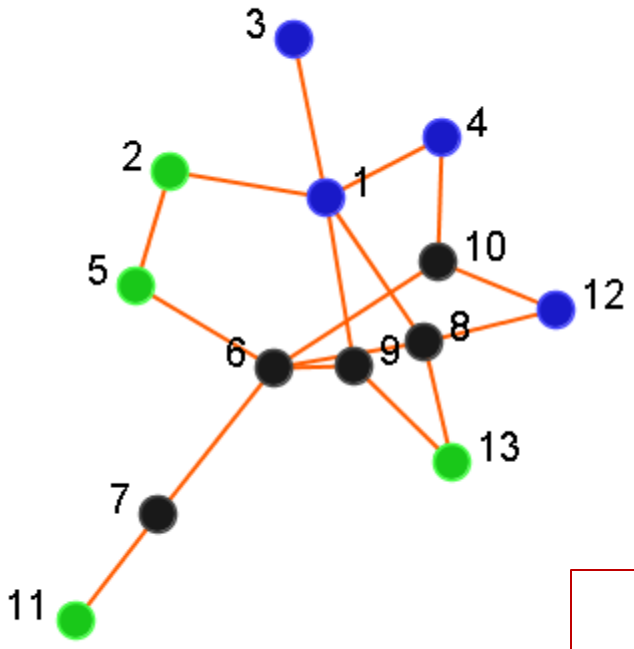


(Nodes colored by Community Membership)

Classification

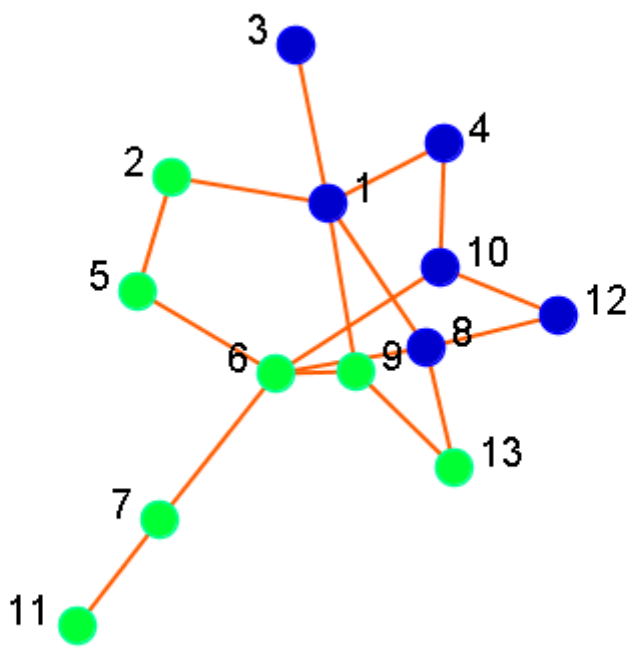
- User Preference or Behavior can be represented as class labels
  - Whether or not clicking on an ad
  - Whether or not interested in certain topics
  - Subscribed to certain political views
  - Like/Dislike a product
- Given
  - A social network
  - Labels of some actors in the network
- Output
  - Labels of remaining actors in the network

Visualization after Prediction



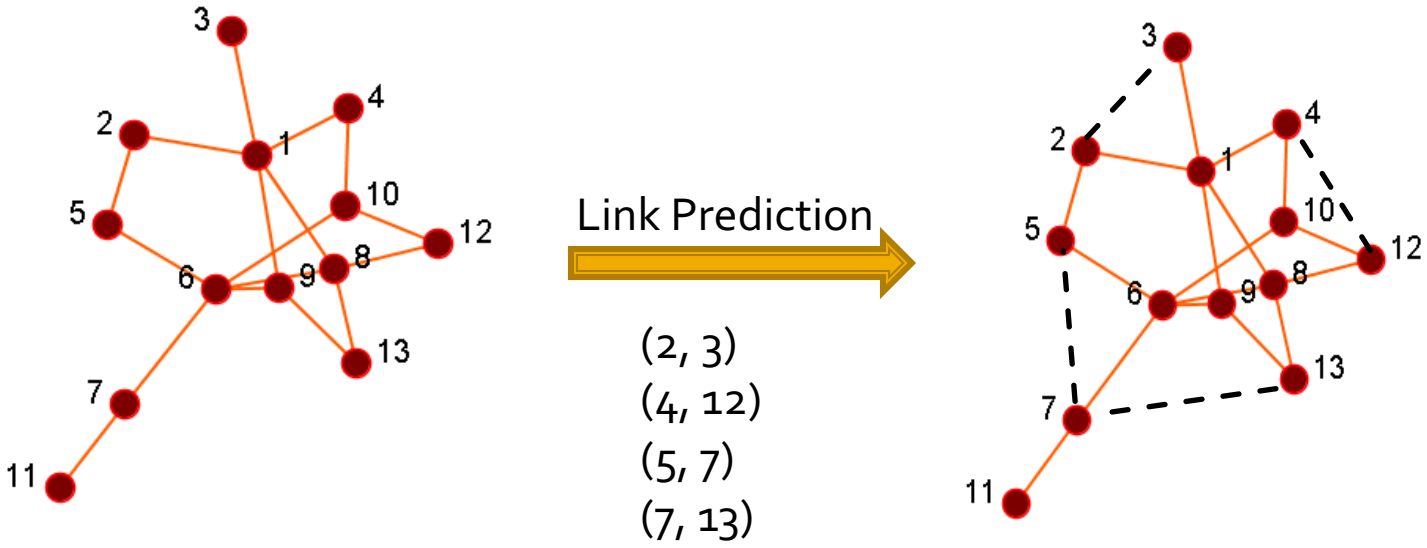
: Smoking
: Non-Smoking
: ? Unknown

Predictions
6: Non-Smoking
7: Non-Smoking
8: Smoking
9: Non-Smoking
10: Smoking



Link Prediction

- Given a social network, predict which nodes are likely to get connected
- Output a list of (ranked) pairs of nodes
- Example: Friend recommendation in Facebook

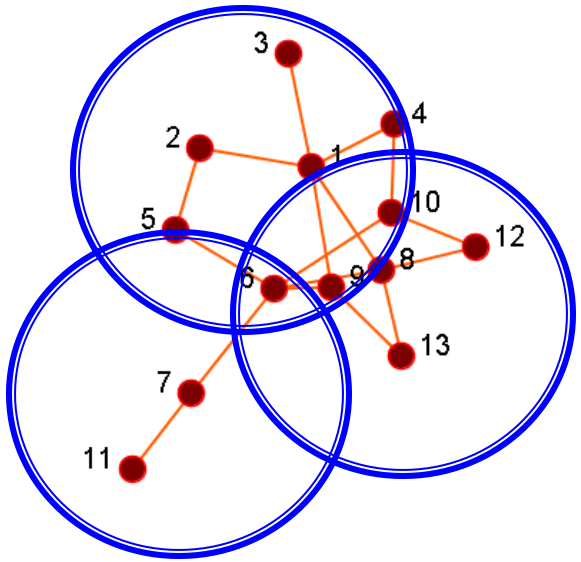


Viral Marketing/Outbreak Detection

- Users have different social capital (or network values) within a social network, hence, how can one make best use of this information?
- Viral Marketing: find out a set of users to provide coupons and promotions to influence other people in the network so my benefit is maximized
- Outbreak Detection: monitor a set of nodes that can help detect outbreaks or interrupt the infection spreading (e.g., H1N1 flu)
- Goal: given a limited budget, how to maximize the overall benefit?

An Example of Viral Marketing

- Find the coverage of the whole network of nodes with the minimum number of nodes
- How to realize it – an example
  - Basic Greedy Selection: Select the node that maximizes the utility, remove the node and then repeat



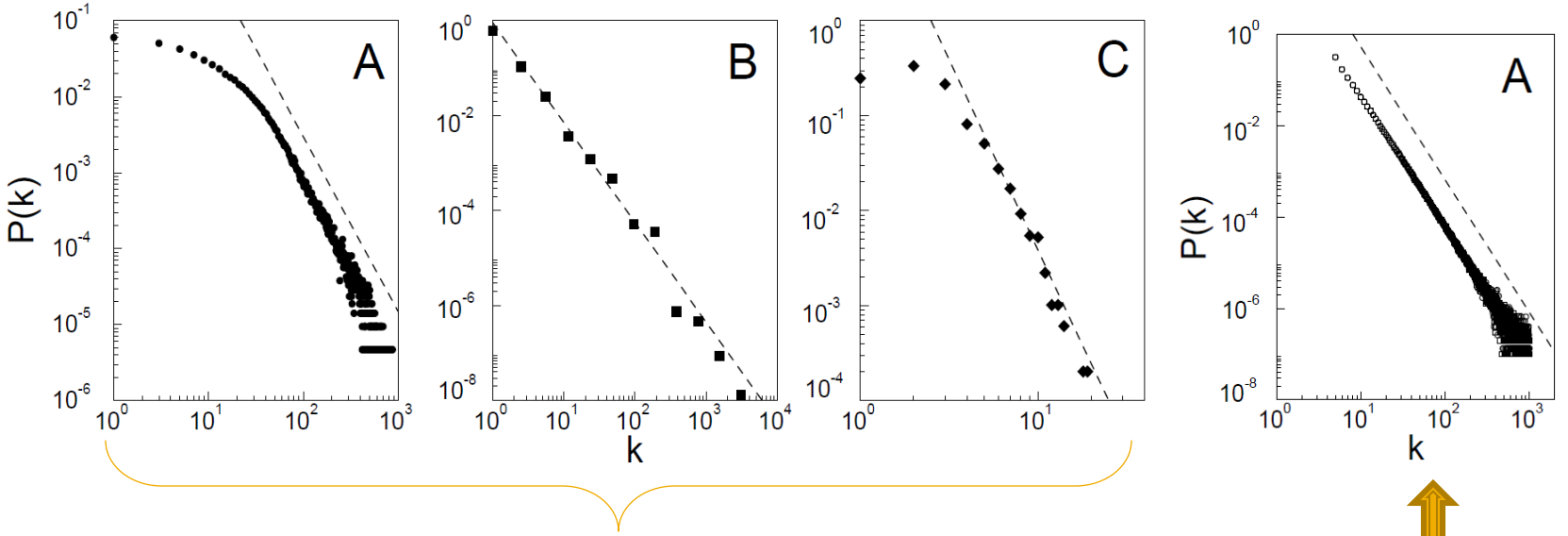
- Select Node 1
- Select Node 8
- Select Node 7

Node 7 is not a node with high centrality!

Network Modeling

- Large Networks demonstrate statistical patterns:
  - Small-world effect (e.g., 6 degrees of separation)
  - Power-law distribution (a.k.a. scale-free distribution)
  - Community structure (high clustering coefficient)
- Model the network dynamics
  - Find a mechanism such that the statistical patterns observed in large-scale networks can be reproduced.
  - Examples: random graph, preferential attachment process
- Used for simulation to understand network properties
  - Thomas Shelling's famous simulation: What could cause the segregation of white and black people
  - Network robustness under attack

Comparing Network Models



observations over various
real-word large-scale networks

outcome of a
network model

(Figures borrowed from "*Emergence of Scaling in Random Networks*")

Social Computing Applications

- Advertizing via Social Networking
- Behavior Modeling and Prediction
- Epidemic Study
- Collaborative Filtering
- Crowd Mood Reader
- Cultural Trend Monitoring
- Visualization
- Health 2.0

PRINCIPLES OF COMMUNITY DETECTION

Communities

- Community: “subsets of actors among whom there are relatively strong, direct, intense, frequent or positive ties.”

  -- Wasserman and Faust, *Social Network Analysis, Methods and Applications*
- Community is a set of actors interacting with each other *frequently*
  - e.g. people attending this conference
- A set of people without interaction is NOT a community
  - e.g. people waiting for a bus at station but don’t talk to each other

- People form communities in Social Media

Example of Communities

Communities from Facebook

Name: **Social Computing**
Type: Organizations
Members: 14 members

Name: **Social Computing**
Type: Internet & Technology
Members: 12 members

Name: **Social Computing Magazine**
Type: Internet & Technology
Members: 34 members

Name: **Trustworthy Social Computing**
Type: Internet & Technology
Members: 28 members

Name: **Social Computing for Business**
Type: Internet & Technology
Members: 421 members

Name: **UCLA Social Sciences Computing**
Type: Internet & Technology
Members: 22 members

Name: **Social Media and Computing**
Type: Organizations
Members: 6 members

Communities from Flickr

! * Urban LIFE in Metropolis ////
4,286 members | 31 discussions | 89,645 items | Created 46 months ago | Join?
UrbanLIFE, People, Parties, Dance, Musik, Life, Love, Culture, Food and Everything what we could imagine by hearing that word URBANLIFE! Have some FUN! Please add... (more)

Islam Is The Way Of Life (Muslim World)
619 members | 13 discussions | 2,685 items | Created 23 months ago | Join?
The word islām is derived from the Arabic verb aslama, which means to accept, surrender or submit. Thus, Islam means submission to and acceptance of God, and believers must... (more)

*** THE CELEBRATION OF ~LIFE~ (Post1~Award1) [only living things]**
4,871 members | 22 discussions | 40,519 items | Created 21 months ago | Join?
WELCOME to THE CELEBRATION OF ~LIFE~ (Post1~Award1) PLEASE INVITE & COMMENT USING only THE CODES FOUND BELOW! ☆☆ This group is for sharing BEAUTIFUL, TOP QUALITY images... (more)

"Enjoy Life!"
2,027 members | 10 discussions | 39,916 items | Created 23 months ago | Join?
There are lovely moments and adorable scenes in our lives. Some are in front of you, and some are just waiting to be discovered. A gaze from someone we love, might touch the... (more)

Baby's life
2,047 members | 185 discussions | 30,302 items | Created 32 months ago | Join?
This group is designed to highlight milestones and important events in your baby's life (ie 1st time smiling/crawling/sitting in a high chair/reading/playing etc). It can also be... (more)

Only group members s
pool

Pond Life
903 members | 20 discussions | 6,877 items | Created 32 months ago | Join?
Pic of the week: chosen from the pool by the group admins. Nuphar by guus timpers Pond Life is a group for all aquatic flora and fauna. Koi ponds, wildlife ponds, garden ponds,... (more)

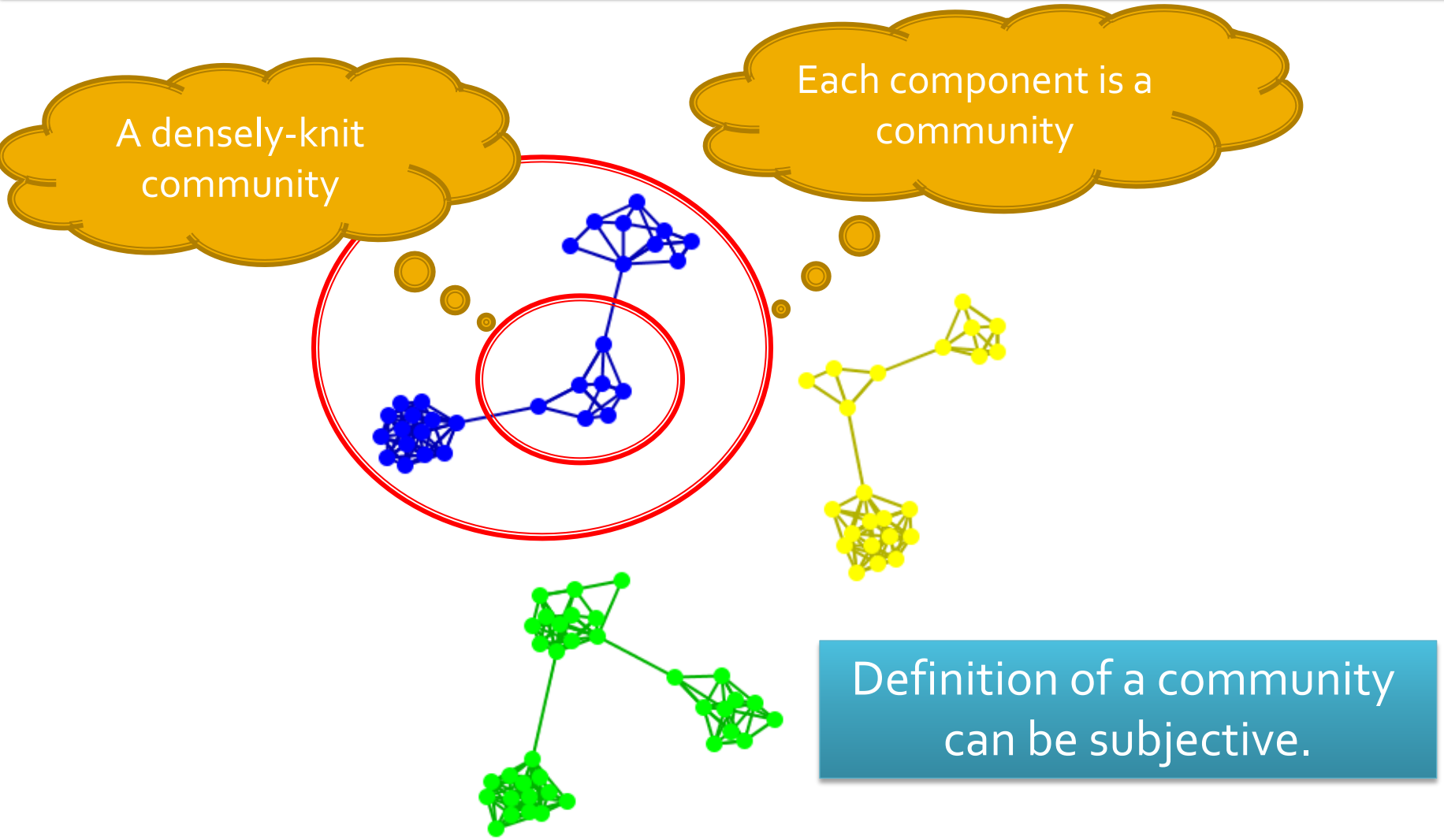
Why Communities in Social Media?

- Human beings are social
- Part of Interactions in social media is a glimpse of the physical world
- People are connected to friends, relatives, and colleagues in the real world as well as online
- Easy-to-use social media allows people to extend their social life in unprecedented ways
  - Difficult to meet friends in the physical world, but much easier to find friend online with similar interests

Community Detection

- Community Detection: “formalize the strong social groups based on the social network properties”
- Some social media sites allow people to join groups, is it necessary to extract groups based on network topology?
  - Not all sites provide community platform
  - Not all people join groups
- Network interaction provides rich information about the relationship between users
  - Groups are *implicitly* formed
  - Can complement other kinds of information
  - Help network visualization and navigation
  - Provide basic information for other tasks

Subjectivity of Community Definition

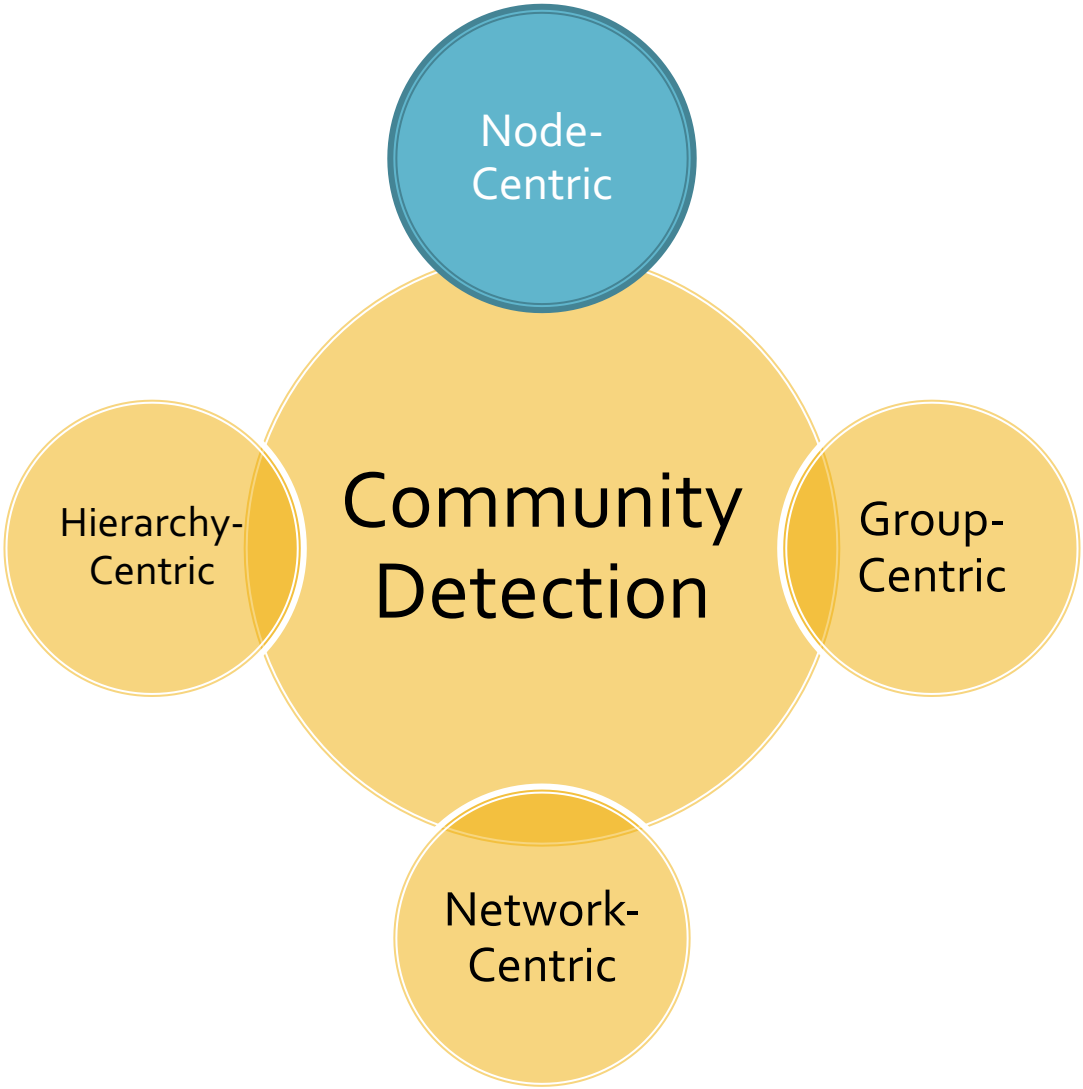


Definition of a community can be subjective.

Taxonomy of Community Criteria

- Criteria vary depending on the tasks
- Roughly, community detection methods can be divided into 4 categories (not exclusive):
- Node-Centric Community
  - Each node in a group satisfies certain properties
- Group-Centric Community
  - Consider the connections within a group as a whole. The group has to satisfy certain properties without zooming into node-level
- Network-Centric Community
  - Partition the whole network into several disjoint sets
- Hierarchy-Centric Community
  - Construct a hierarchical structure of communities

Node-Centric Community Detection

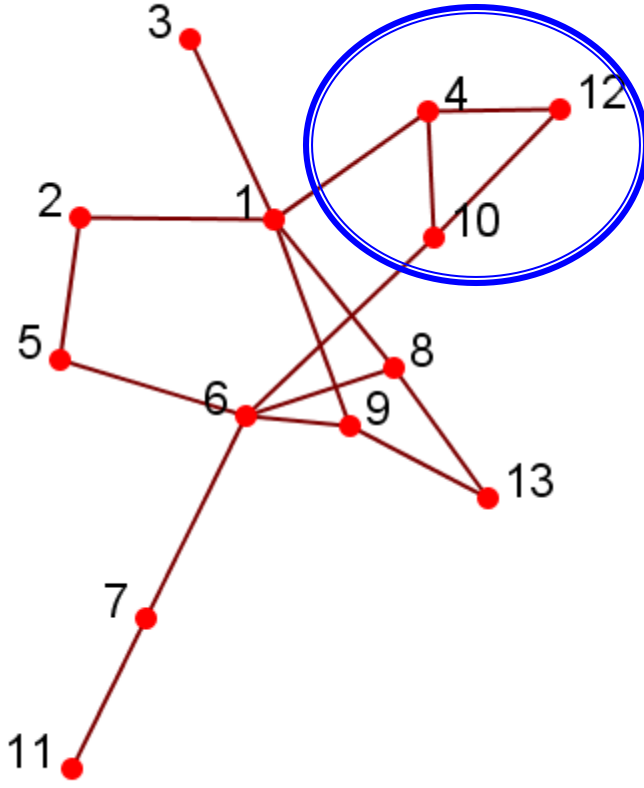


Node-Centric Community Detection

- Nodes satisfy different properties
  - Complete Mutuality
    - cliques
  - Reachability of members
    - k-clique, k-clan, k-club
  - Nodal degrees
    - k-plex, k-core
  - Relative frequency of Within-Outside Ties
    - LS sets, Lambda sets
- Commonly used in traditional social network analysis
- Here, we discuss some representative ones

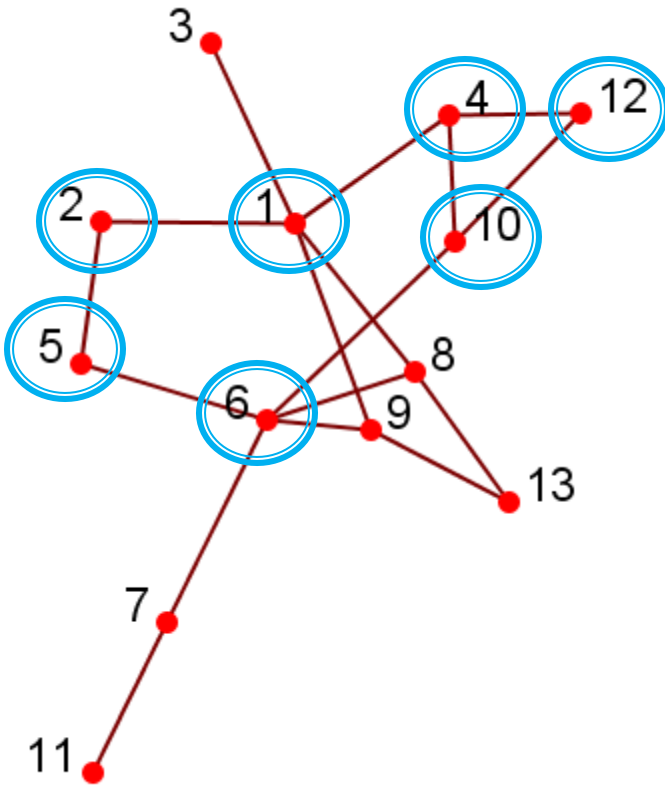
Complete Mutuality: Clique

- A maximal complete subgraph of three or more nodes all of which are adjacent to each other
- NP-hard to find the maximal clique
- Recursive pruning: To find a clique of size k, remove those nodes with less than k-1 degrees
- Very strict definition, unstable
- Normally use cliques as a core or seed to explore larger communities



Geodesic

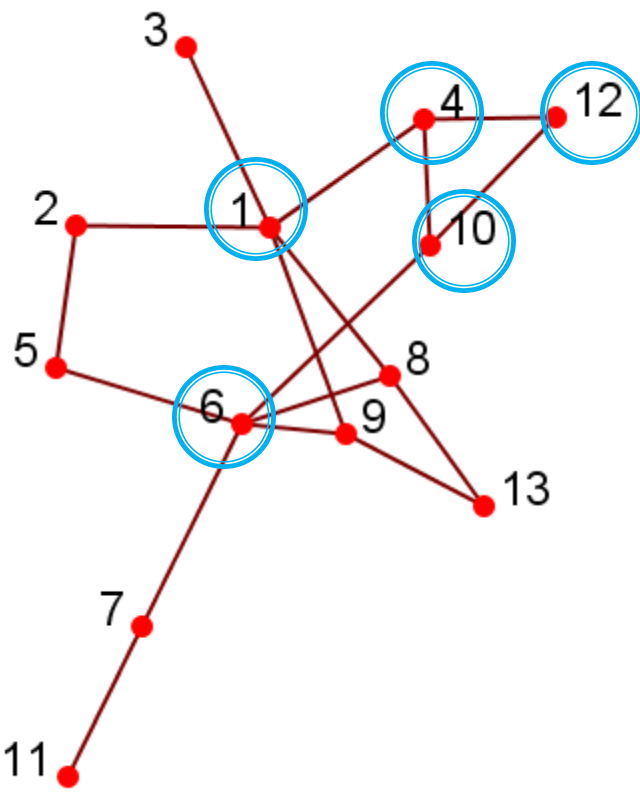
- Reachability is calibrated by the Geodesic distance
- Geodesic: a shortest path between two nodes (12 and 6)
  - Two paths: 12-4-1-2-5-6, 12-10-6
  - 12-10-6 is a geodesic
- Geodesic distance: #hops in geodesic between two nodes
  - e.g., d(12, 6) = 2, d(3, 11)=5
- Diameter: the maximal geodesic distance for any 2 nodes in a network
  - #hops of the longest shortest path



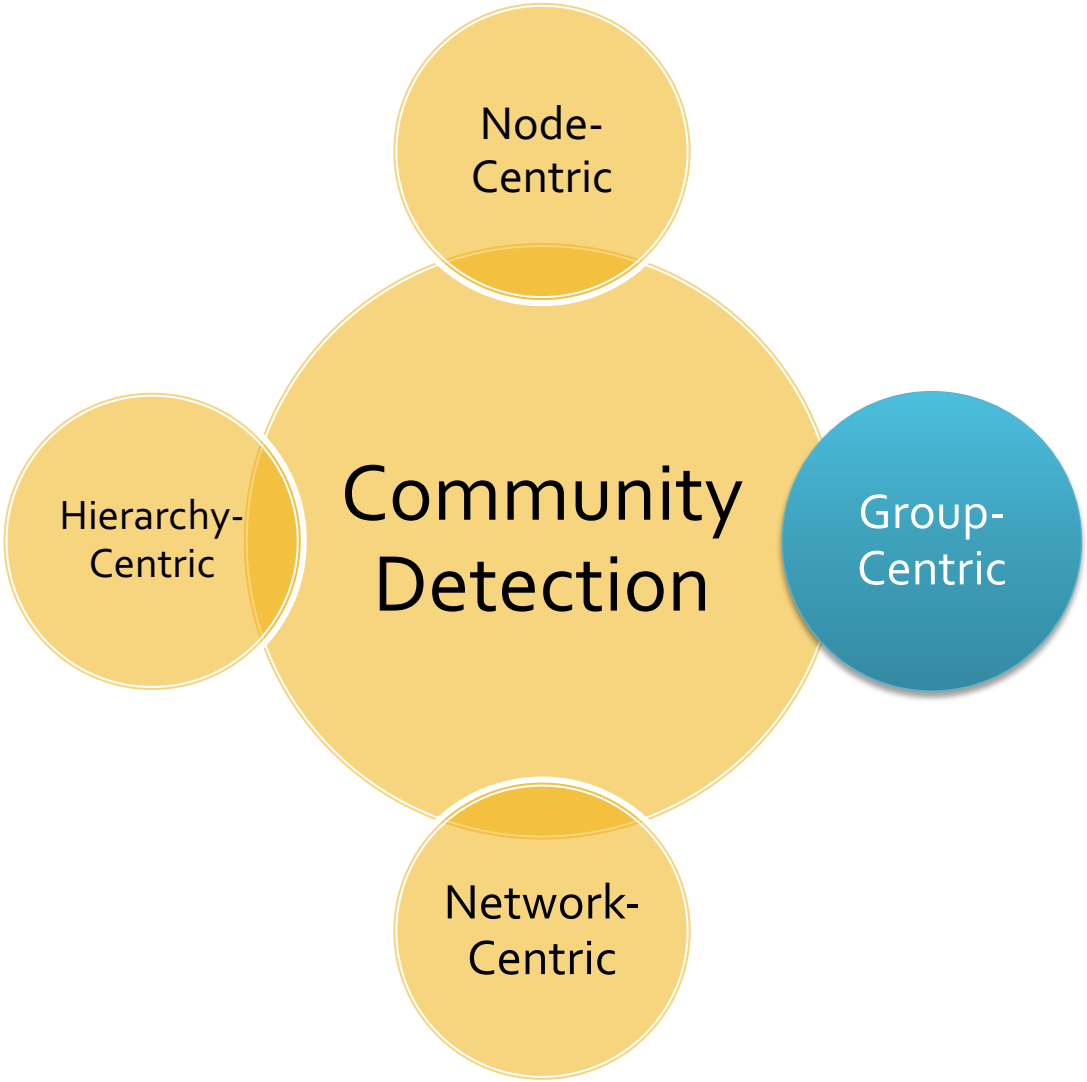
Diameter = 5

Reachability: k-clique, k-club

- Any node in a group should be reachable in k hops
- k-clique: a maximal subgraph in which the largest geodesic distance between any nodes <= k
- A k-clique can have diameter larger than k within the subgraph
  - e.g., 2-clique {12, 4, 10, 1, 6}
  - Within the subgraph d(1, 6) = 3
- k-club: a substructure of diameter <= k
  - e.g., {1,2,5,6,8,9}, {12, 4, 10, 1} are 2-clubs



Group-Centric Community Detection



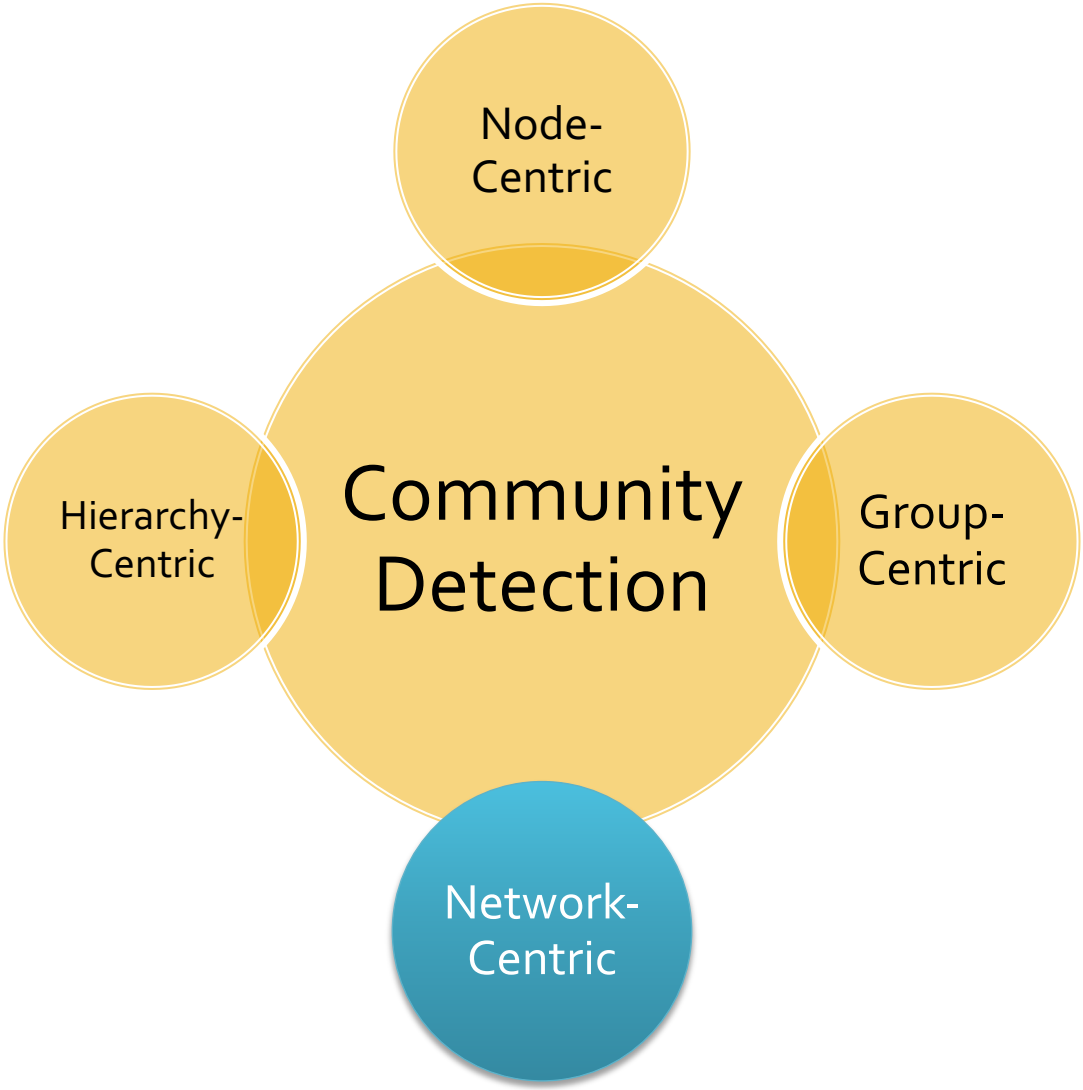
Group-Centric Community Detection

- Consider the connections within a group as whole,
- OK for some nodes to have low connectivity
- A subgraph with V_s nodes and E_s edges is a γ-dense quasi-clique if

$$\frac{E_s}{V_s(V_s-1)/2} \geq \gamma$$

- Recursive pruning:
  - Sample a subgraph, find a maximal γ-dense quasi-clique (the resultant size = k)
  - Remove the nodes that
    - whose degree < kγ
    - all their neighbors with degree < kγ

Network-Centric Community Detection



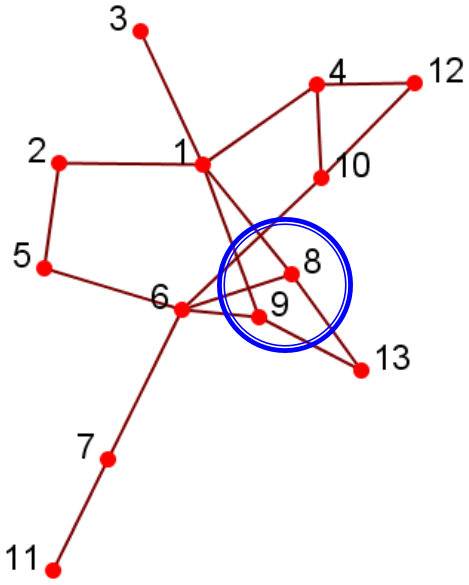
Network-Centric Community Detection

- To form a group, we need to consider the connections of the nodes globally.

- Goal: partition the network into disjoint sets
  - Groups based on Node Similarity
  - Groups based on Latent Space Model
  - Groups based on Block Model Approximation
  - Groups based on Cut Minimization
  - Groups based on Modularity Maximization

Node Similarity

- Node similarity is defined by how similar their interaction patterns are
- Two nodes are structurally equivalent if they connect to the same set of actors
  - e.g., nodes 8 and 9 are structurally equivalent
- Groups are defined over equivalent nodes
  - Too strict
  - Rarely occur in a large-scale
  - Relaxed equivalence class is difficult to compute
- In practice, use vector similarity
  - e.g., cosine similarity, Jaccard similarity



Vector Similarity

a vector

structurally equivalent

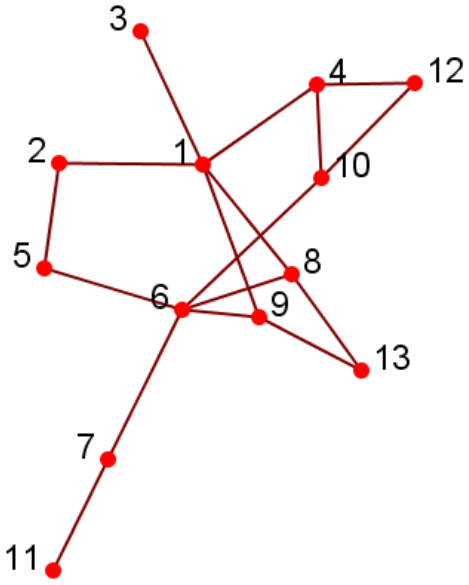
	1	2	3	4	5	6	7	8	9	10	11	12	13
5		1				1							
8	1					1							1
9	1					1							1

Cosine Similarity: $\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}.$

$$sim(5,8) = \frac{1}{\sqrt{2} \times \sqrt{3}} = \frac{1}{\sqrt{6}}$$

Jaccard Similarity: $J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$

$$J(5,8) = \frac{|\{6\}|}{|\{1,2,6,13\}|} = 1/4$$



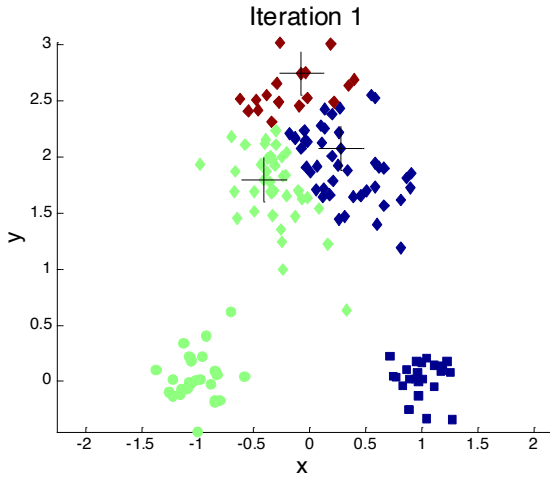
Clustering based on Node Similarity

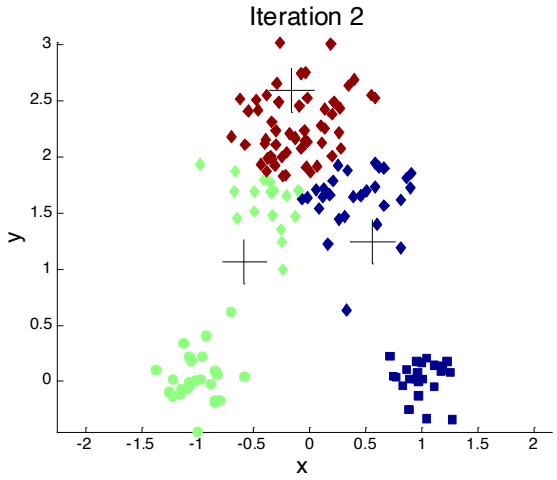
- For practical use with huge networks:
  - Consider the connections as features
  - Use Cosine or Jaccard similarity to compute vertex similarity
  - Apply classical k-means clustering Algorithm
- K-means Clustering Algorithm
  - Each cluster is associated with a centroid (center point)
  - Each node is assigned to the cluster with the closest centroid

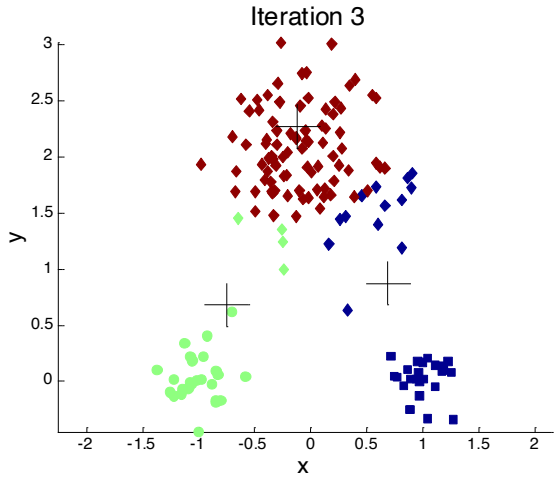
Algorithm 1 Basic K-means Algorithm.

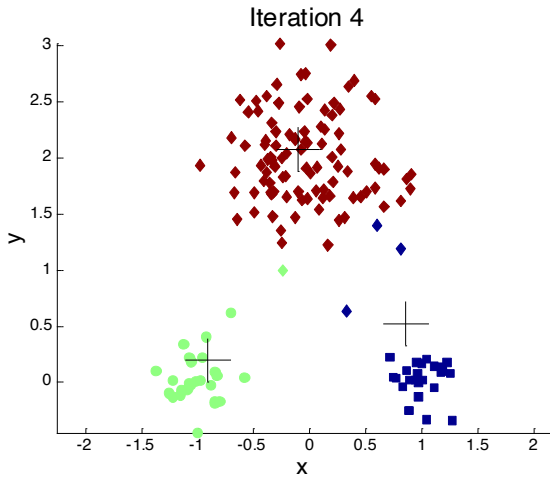
1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

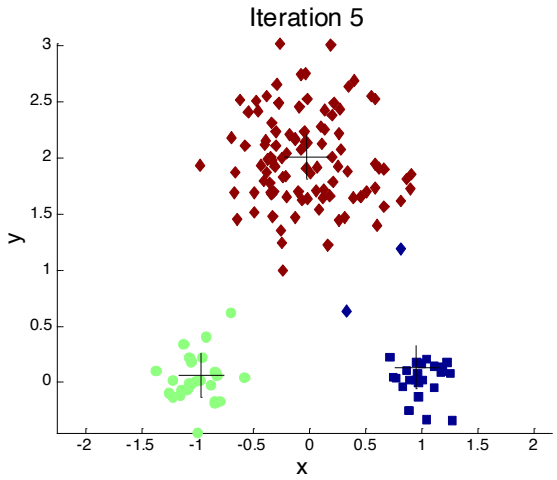
Illustration of k-means clustering

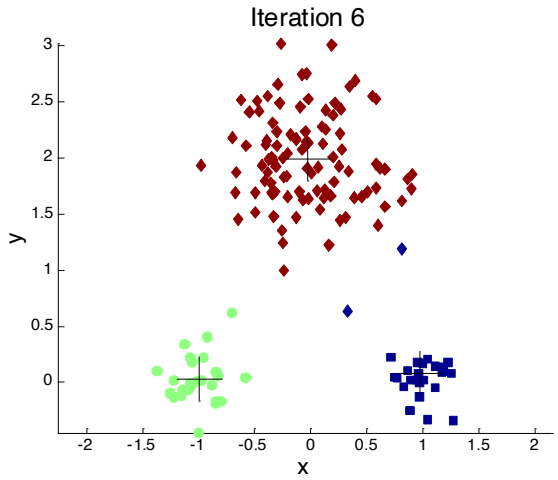












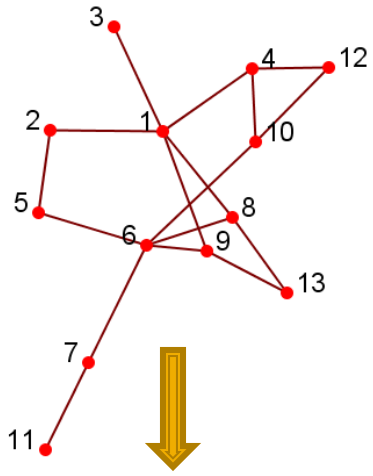
Groups on Latent-Space Models

- Latent-space models: Transform the nodes in a network into a lower-dimensional space such that the distance or similarity between nodes are kept in the Euclidean space
- Multidimensional Scaling (MDS)
  - Given a network, construct a proximity matrix to denote the distance between nodes (e.g. geodesic distance)
  - Let D denotes the *square distance* between nodes
  - $S \in R^{n \times k}$ denotes the coordinates in the lower-dimensional space

$$SS^T = -\frac{1}{2}(I - \frac{1}{n}ee^T)D(I - \frac{1}{n}ee^T) = \Delta(D)$$

  - Objective: minimize the difference $\min \| \Delta(D) - SS^T \|_F$
  - Let $\Lambda = diag(\lambda_1, \cdots, \lambda_k)$:op-k eigenvalues of), V tl Δ top-k eigenvectors
  - Solution: $S = V\Lambda^{1/2}$
- Apply k-means to S to obtain clusters

MDS-example



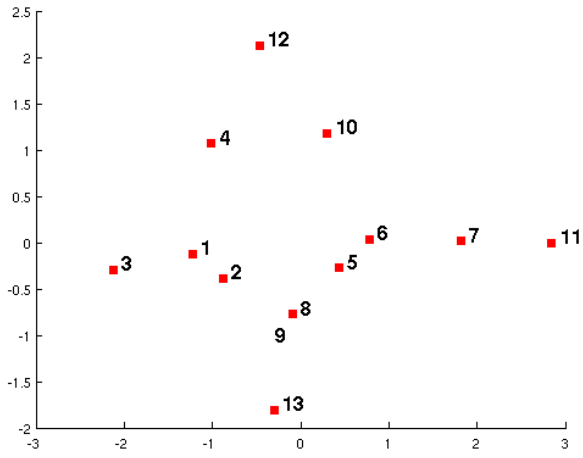
Geodesic Distance Matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	1	1	1	2	2	3	1	1	2	4	2	2
2	1	0	2	2	1	2	3	2	2	3	4	3	3
3	1	2	0	2	3	3	4	2	2	3	5	3	3
4	1	2	2	0	3	2	3	2	2	1	4	1	3
5	2	1	3	3	0	1	2	2	2	2	3	3	3
6	2	2	3	2	1	0	1	1	1	1	2	2	2
7	3	3	4	3	2	1	0	2	2	2	1	3	3
8	1	2	2	2	2	1	2	0	2	2	3	3	1
9	1	2	2	2	2	1	2	2	0	2	3	3	1
10	2	3	3	1	2	1	2	2	2	0	3	1	3
11	4	4	5	4	3	2	1	3	3	3	0	4	4
12	2	3	3	1	3	2	3	3	3	1	4	0	4
13	2	3	3	3	3	2	3	1	1	3	4	4	0

MDS

S

-1.22	-0.12
-0.88	-0.39
-2.12	-0.29
-1.01	1.07
0.43	-0.28
0.78	0.04
1.81	0.02
-0.09	-0.77
-0.09	-0.77
0.30	1.18
2.85	0.00
-0.47	2.13
-0.29	-1.81

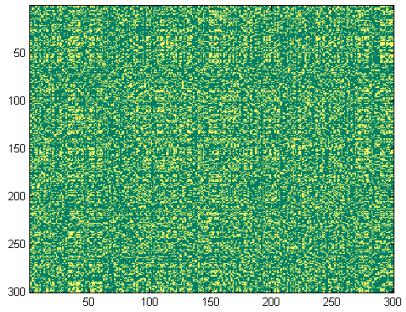


k-means

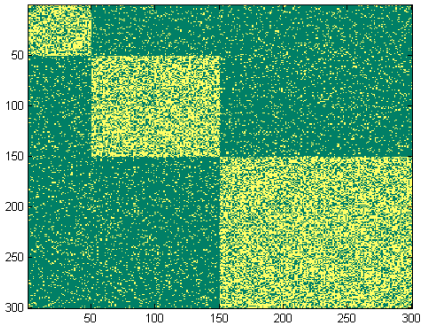
1, 2, 3, 4, 10, 12

5, 6, 7, 8, 9, 11, 13

Block-Model Approximation



After Reordering



Network Interaction Matrix

Block Structure

➢Objective: Minimize the difference between an interaction matrix and a block structure

$$\min_{S,\Sigma} \|A - S\Sigma S^T\|_F$$

$$s.t. \quad S \in \{0,1\}^{n\times k}, \Sigma \in R^{k\times k} \text{ is diagonal}$$

S is a community indicator matrix

➢Challenge: S is discrete, difficult to solve

➢Relaxation: Allow S to be continuous satisfying $S^T S = I_k$

➢Solution: the top eigenvectors of A

➢Post-Processing: Apply k-means to S to find the partition

Cut-Minimization

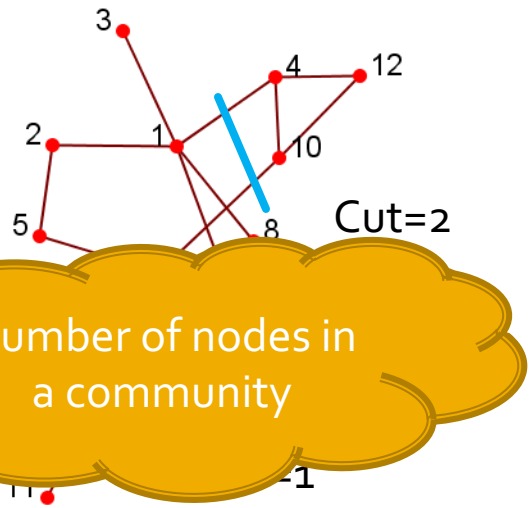
- Between-group interactions should be infrequent
- Cut: number of edges between two sets of nodes
- Objective: minimize the cut

$$cut(C_1, C_2, \cdots, C_k) = \sum_{i=1}^{k} cut(C_i, \overline{C_i})$$

  - Limitations: often find communities of only one node
  - Need to consider the group size
- Two commonly-used variants:

$$\text{Ratio-cut}(C_1, C_2, \cdots, C_k) = \sum_{i=1}^{k} \frac{cut(C_i, \overline{C_i})}{|V_i|}$$

$$\text{Normalized-cut}(C_1, C_2, \cdots, C_k) = \sum_{i=1}^{k} \frac{cut(C_i, \overline{C_i})}{vol(V_i)}$$



Graph Laplacian

- Can be relaxed into the following min-trace problem

- L is the (normalized) Graph Laplacian

$$\min_{S \in R^{n \times k}} Tr(S^T L S) \quad s.t.\ S^T S = I$$

$$\begin{aligned} L &= D - A \\ \text{normalized-}L &= I - D^{-1/2} A D^{-1/2} \end{aligned}$$

$$D = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & d_n \end{pmatrix}$$

- Solution: S are the eigenvectors of L with eigenvalues (except the first one)
- Post-Processing: apply k-means to S
- a.k.a.Spectral Clustering

Modularity Maximization

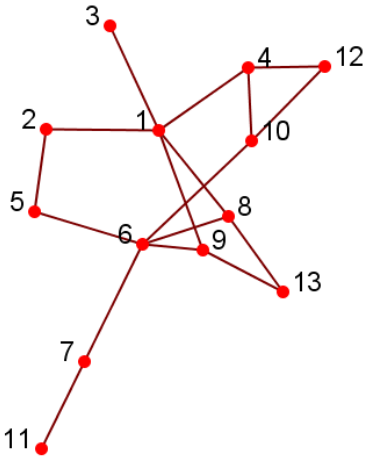
- Modularity measures the group interactions compared with the expected random connections in the group
- In a network with m edges, for two nodes with degree d_i and d_j, the expected random connections between them are
- The interaction utility in a group:

$$d_i d_j / 2m$$

- To partition the group into multiple groups, v

$$\sum_{i \in C, j \in C} A_{ij} - d_i d_j / 2m$$

$$\max \quad \frac{1}{2m} \sum_{C} \sum_{i \in C, j \in C} A_{ij} - d_i d_j / 2m$$



Expected Number of edges between 6 and 9 is 5*3/(2*17) = 15/34

Modularity Matrix

- The modularity maximization can also be formulated in matrix form

$$Q = \frac{1}{2m} Tr(S^T B S)$$

- B is the modularity matrix

$$B_{ij} = A_{ij} - d_i d_j / 2m$$

- Solution: top eigenvectors of the modularity matrix

Matrix Factorization Form

- For latent space models, block models, spectral clustering and modularity maximization
- All can be formulated as

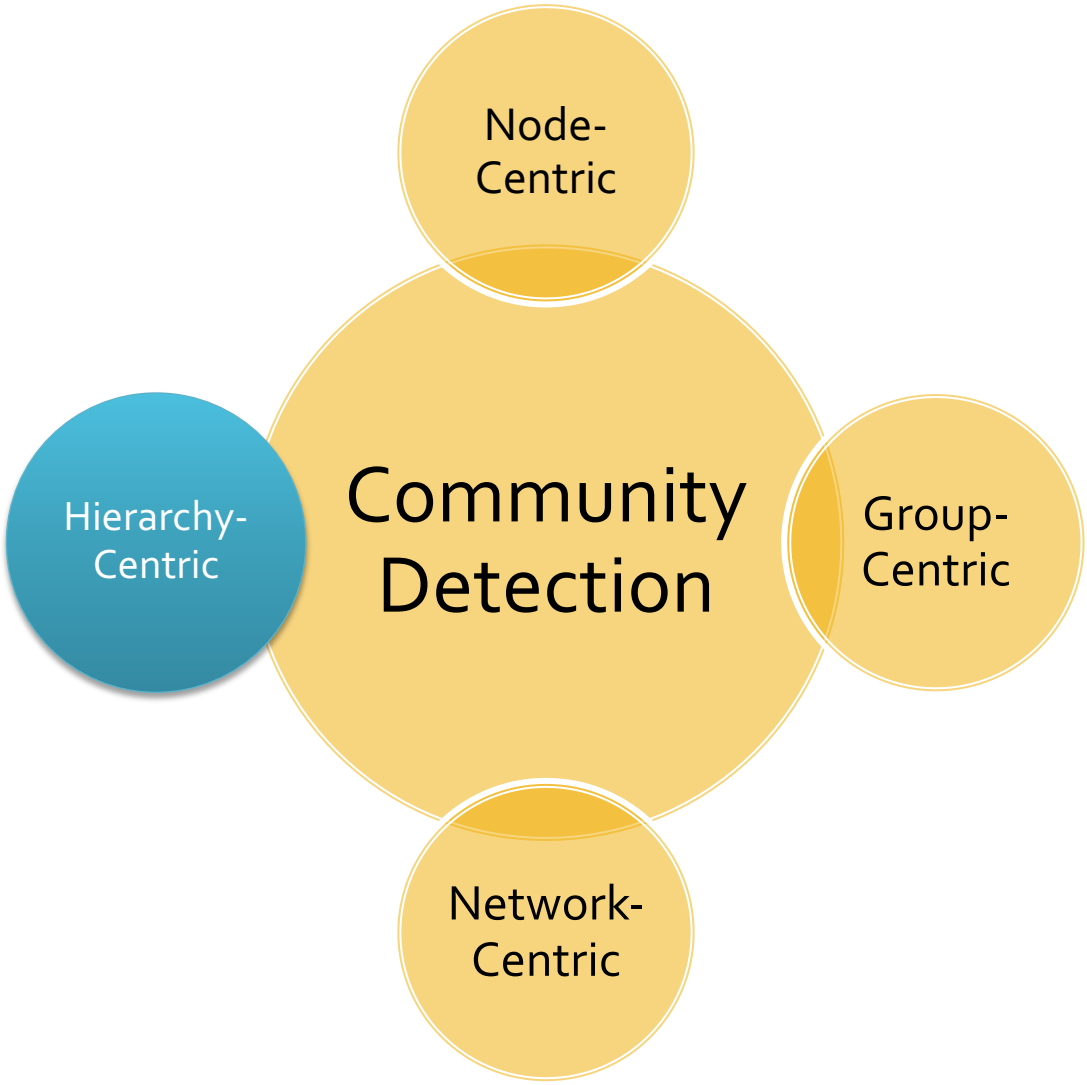
$$\max(\min)_S \quad Tr(S^T X S)$$
$$s.t. \quad S^T S = I$$

X= $\Delta(D)$ (Latent Space Models)
Sociomatrix (Block Model Approximation)
Graph Laplacian (Cut Minimization)
Modularity Matrix (Modularity maximization)

Recap of Network-Centric Community

- Network-Centric Community Detection
  - Groups based on Node Similarity
  - Groups based on Latent Space Models
  - Groups based on Cut Minimization
  - Groups based on Block-Model Approximation
  - Groups based on Modularity maximization
- Goal: Partition network nodes into several disjoint sets
- Limitation: Require the user to specify the number of communities beforehand

Hierarchy-Centric Community Detection

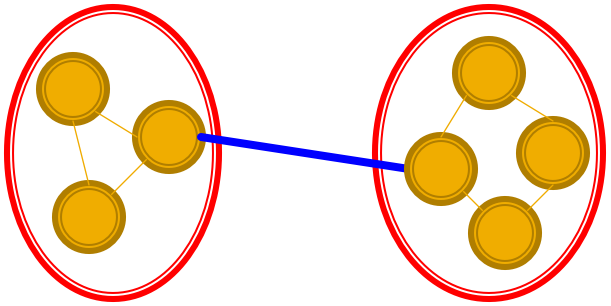


Hierarchy-Centric Community Detection

- Goal: Build a hierarchical structure of communities based on network topology
- Facilitate the analysis at different resolutions
- Representative Approaches:
  - Divisive Hierarchical Clustering
  - Agglomerative Hierarchical Clustering

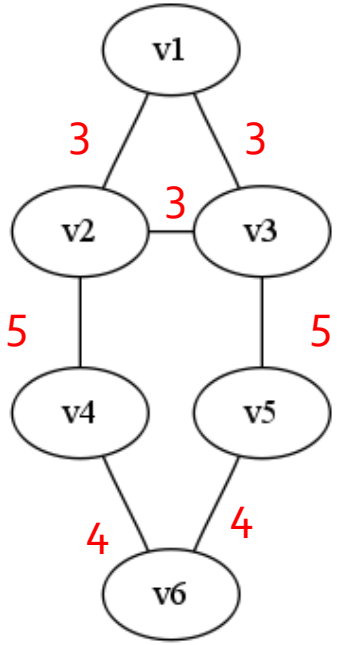
Divisive Hierarchical Clustering

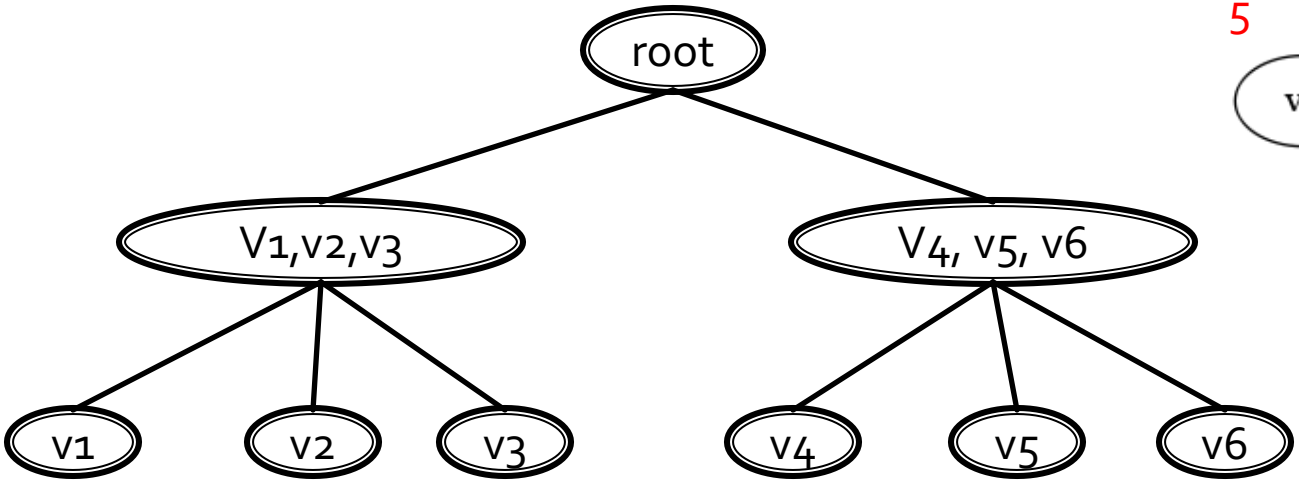
- Divisive Hierarchical Clustering
  - Partition the nodes into several sets
  - Each set is further partitioned into smaller sets
- Network-centric methods can be applied for partition
- One particular example is based on edge-betweenness
- Edge-Betweenness: Number of shortest paths between any pair of nodes that pass through the edge
- Between-group edges tend to have larger edge-betweenness



Divisive clustering on Edge-Betweenness

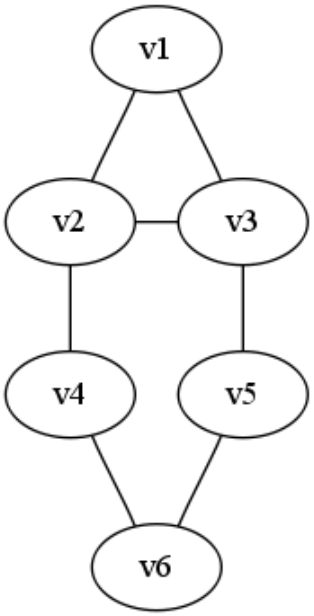
- Progressively remove edges with the highest betweenness
  - Remove e(2,4), e(3, 5)
  - Remove e(4,6), e(5,6)
  - Remove e(1,2), e(2,3), e(3,1)

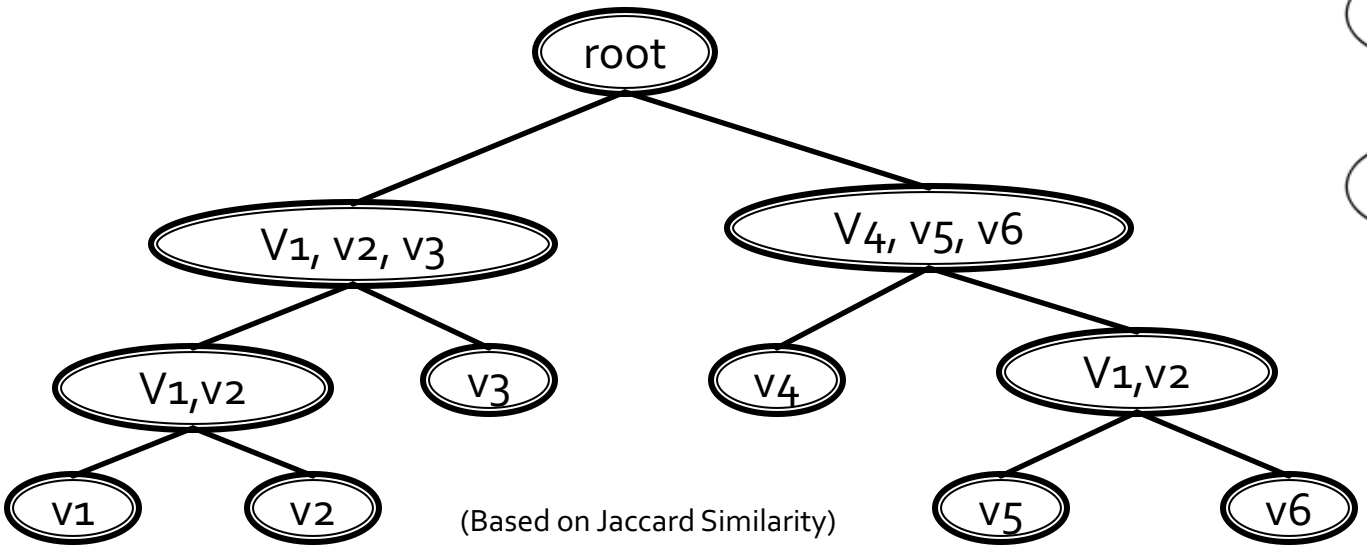




Agglomerative Hierarchical Clustering

- Initialize each node as a community
- Choose two communities satisfying certain criteria and merge them into larger ones
  - Maximum Modularity Increase
  - Maximum Node Similarity

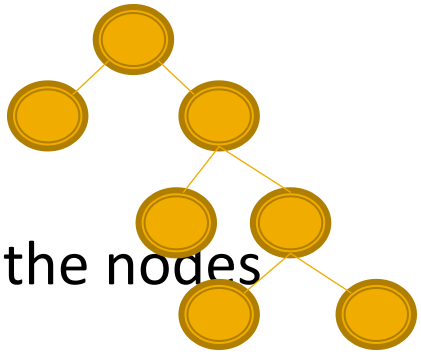




(Based on Jaccard Similarity)

Recap of Hierarchical Clustering

- Most hierarchical clustering algorithm output a binary tree
  - Each node has two children nodes
  - Might be highly imbalanced



- Agglomerative clustering can be very sensitive to the nodes processing order and merging criteria adopted.

- Divisive clustering is more stable, but generally more computationally expensive

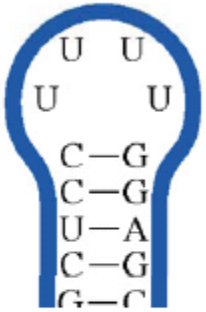
Summary of Community Detection

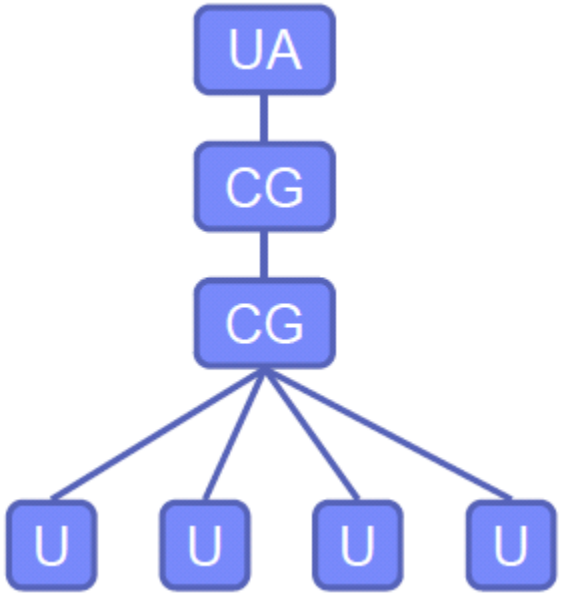
- The Optimal Method?
- It varies depending on applications, networks, computational resources etc.
- Scalability can be a concern for networks in social media
- Other lines of research
  - Communities in directed networks
  - Overlapping communities
  - Community evolution
  - Group profiling and interpretation

Graph Data Mining

- DNA sequence

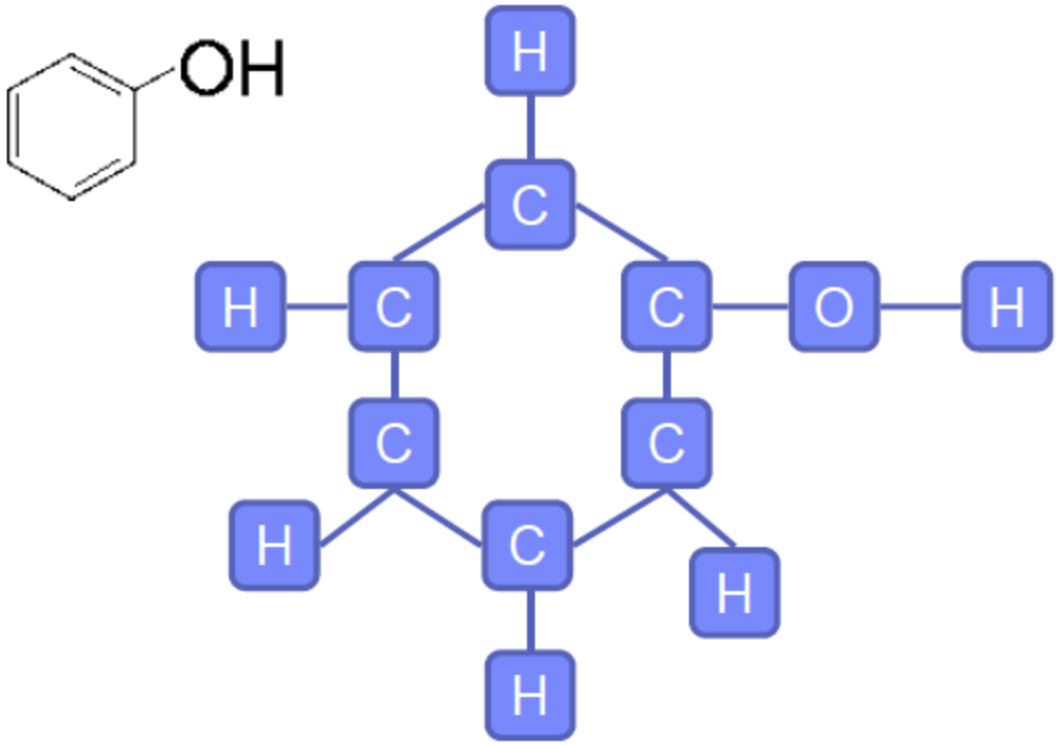
- RNA





Graph Data Mining

- Compounds



- 

Outline

- Graph Pattern Mining
  - Mining Frequent Subgraph Patterns
  - Graph Indexing
  - Graph Similarity Search
- Graph Classification
  - Graph pattern-based approach
  - Machine Learning approaches
- Graph Clustering
  - Link-density-based approach

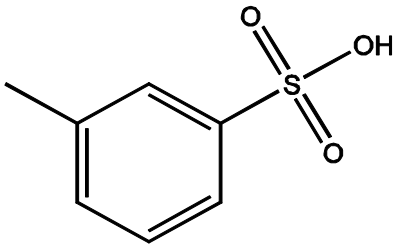
Graph Pattern Mining

- *Frequent* subgraphs
  - A (sub)graph is ***frequent*** if its *support* (occurrence frequency) in a given dataset is no less than a *minimum support* threshold
- Support of a graph g is defined as the percentage of graphs in G which have g as subgraph

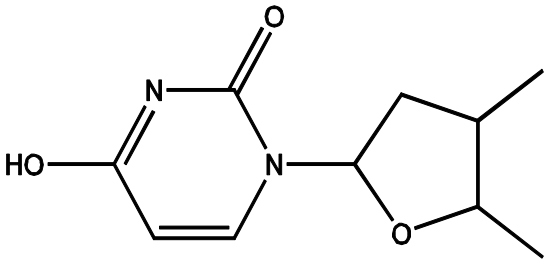
- Applications of graph pattern mining
  - Mining biochemical structures
  - Program control flow analysis
  - Mining XML structures or Web communities
  - Building blocks for graph classification, clustering, compression, comparison, and correlation analysis

Example: Frequent Subgraphs

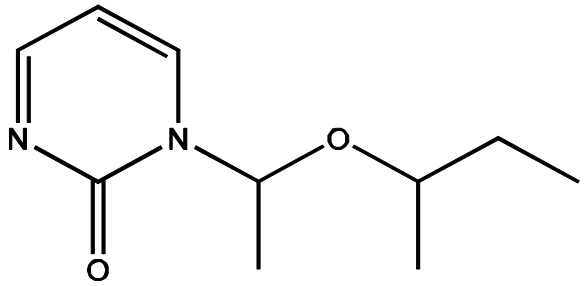
GRAPH DATASET



(A)



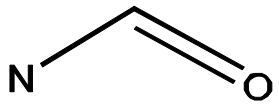
(B)



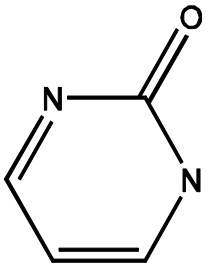
(C)

FREQUENT PATTERNS (MIN SUPPORT IS 2)

(1)

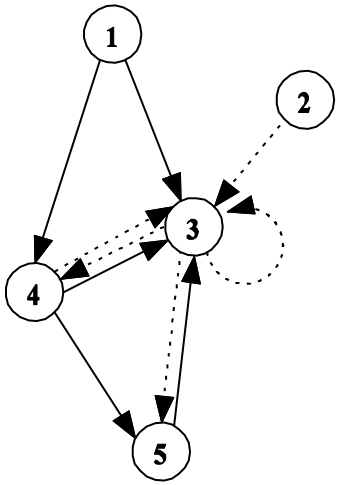


(2)

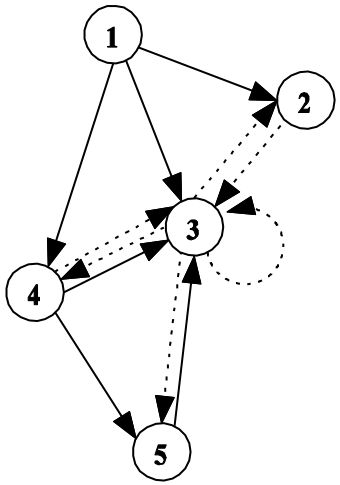


Example

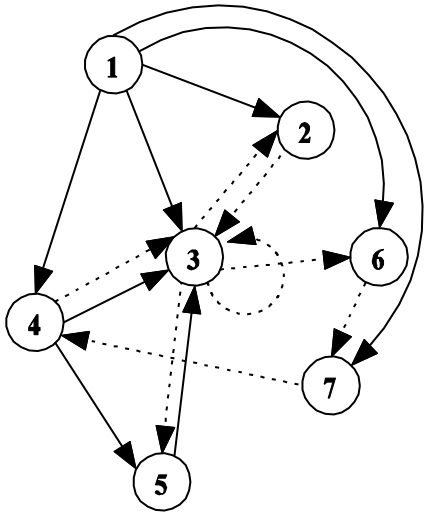
GRAPH DATASET



(1)



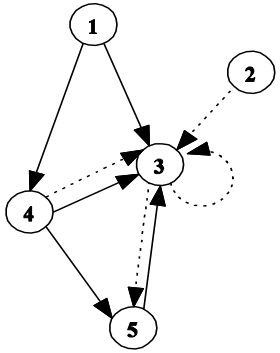
(2)



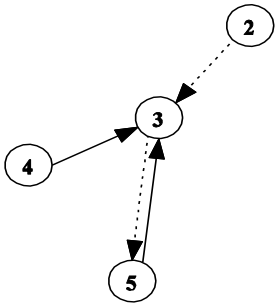
(3)

1: makepat
2: esc
3: addstr
4: getccl
5: dodash
6: in_set_2
7: stclose

FREQUENT PATTERNS (MIN SUPPORT IS 2)



(1)



(2)

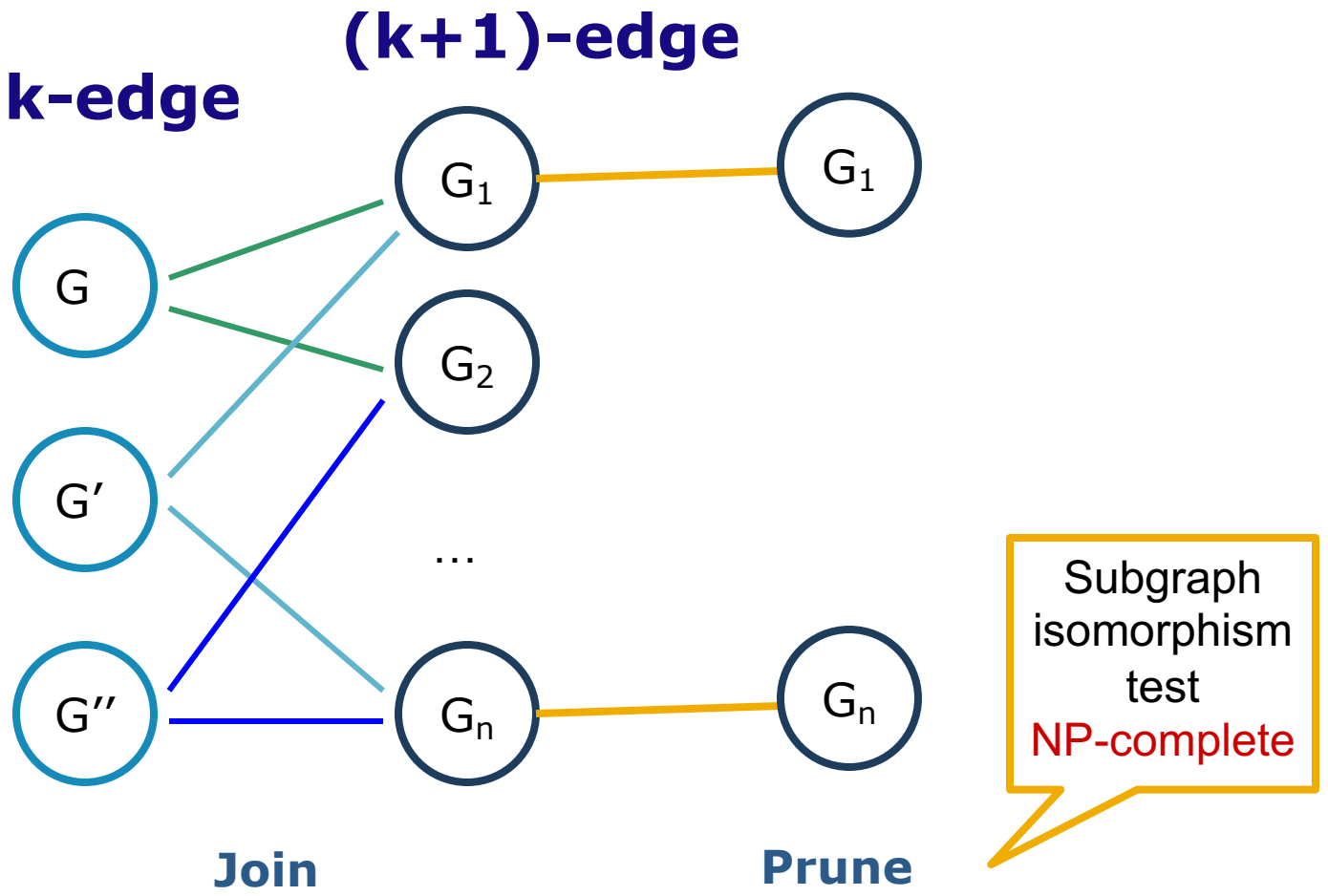
Graph Mining Algorithms

- Incomplete beam search – Greedy (Subdue)
- Inductive logic programming (WARMR)
- Graph theory-based approaches
  - Apriori-based approach
  - Pattern-growth approach

Properties of Graph Mining Algorithms

- Search order
  - breadth vs. depth
- Generation of candidate subgraphs
  - apriori vs. pattern growth
- Elimination of duplicate subgraphs
  - passive vs. active
- Support calculation
  - embedding store or not
- Discover order of patterns
  - path → tree → graph

Apriori-Based Approach

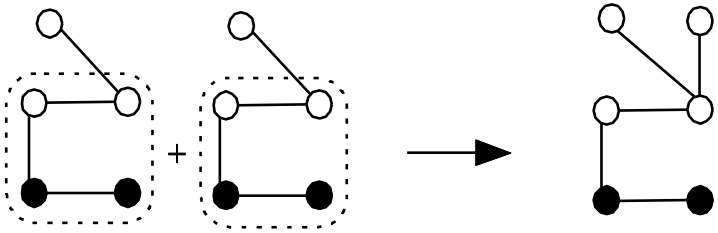


Example

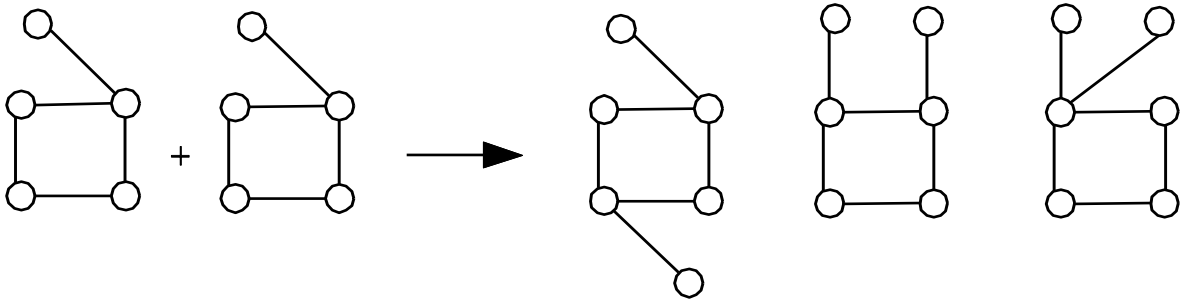
交易ID	商品ID列表
T100	I1，I2，I5
T200	I2，I4
T300	I2，I3
T400	I1，I2，I4
T500	I1，I3
T600	I2，I3
T700	I1，I3
T800	I1，I2，I3，I5
T900	I1，I2，I3

Apriori-Based, Breadth-First Search

- Methodology: breadth-search, joining two graphs

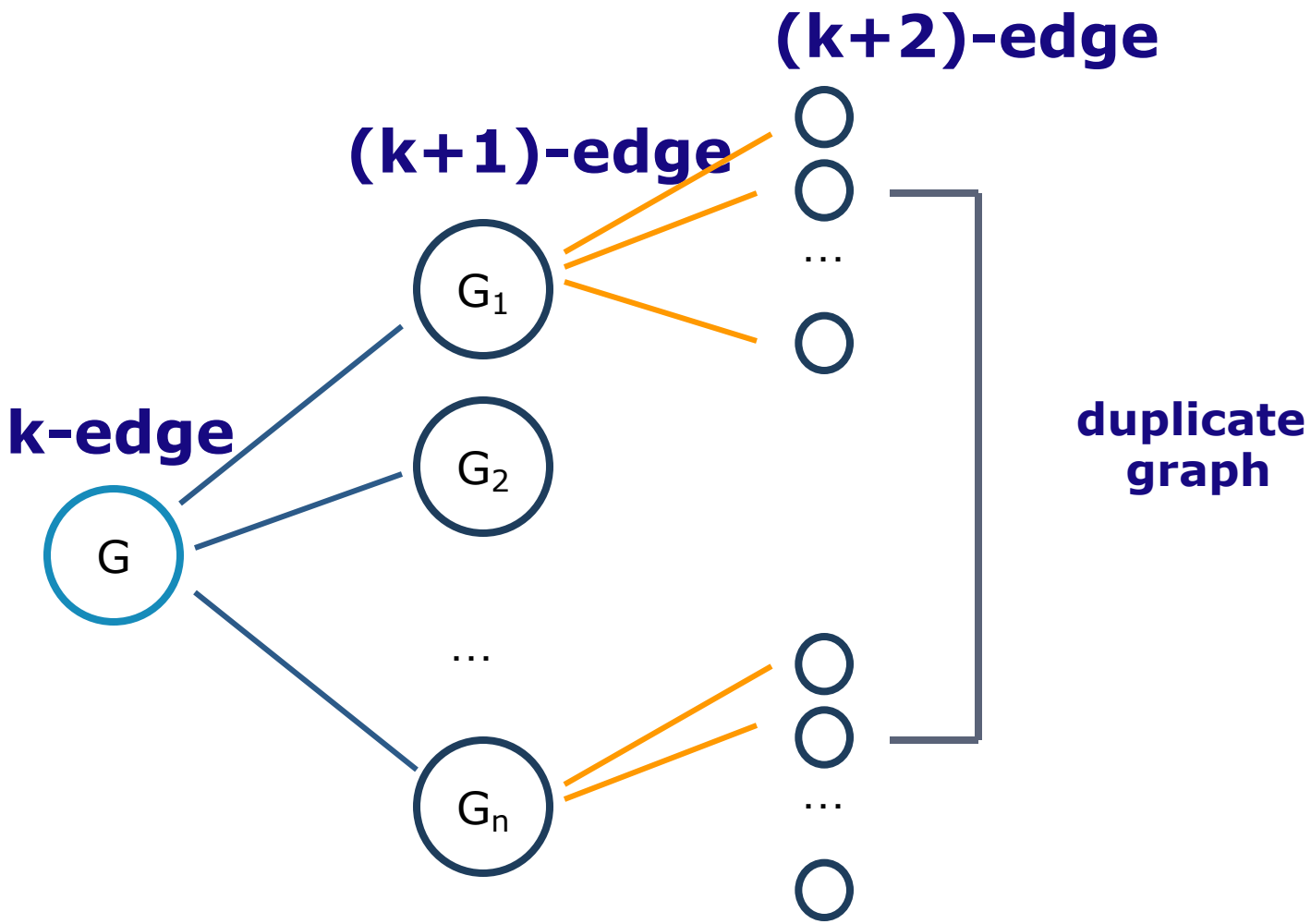


- AGM (Inokuchi, et al.)
  - generates new graphs with one more node



- FSG (Kuramochi and Karypis)
  - generates new graphs with one more edge

Pattern Growth Method



Graph Pattern Explosion Problem

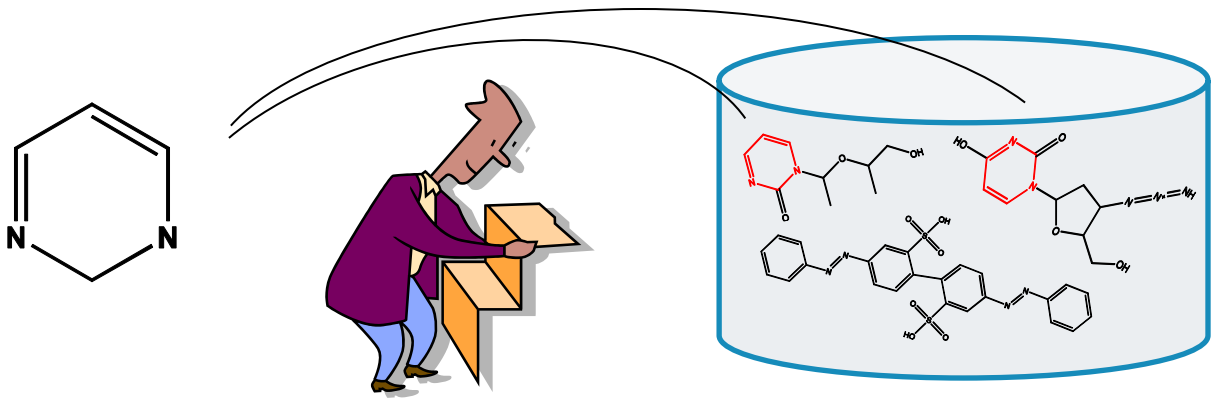
- If a graph is frequent, all of its subgraphs are frequent
  - **the Apriori property**
- An **n**-edge frequent graph may have 2^n subgraphs
- Among **422** chemical compounds which are confirmed to be active in an AIDS antiviral screen dataset,
  - there are **1,000,000** frequent graph patterns if the minimum support is 5%

Closed Frequent Graphs

- A frequent graph G is closed
  - if there exists no supergraph of G that carries the same support as G
- If some of G's subgraphs have the same support
  - it is unnecessary to output these subgraphs
  - nonclosed graphs
- Lossless compression
  - Still ensures that the mining result is complete

Graph Search

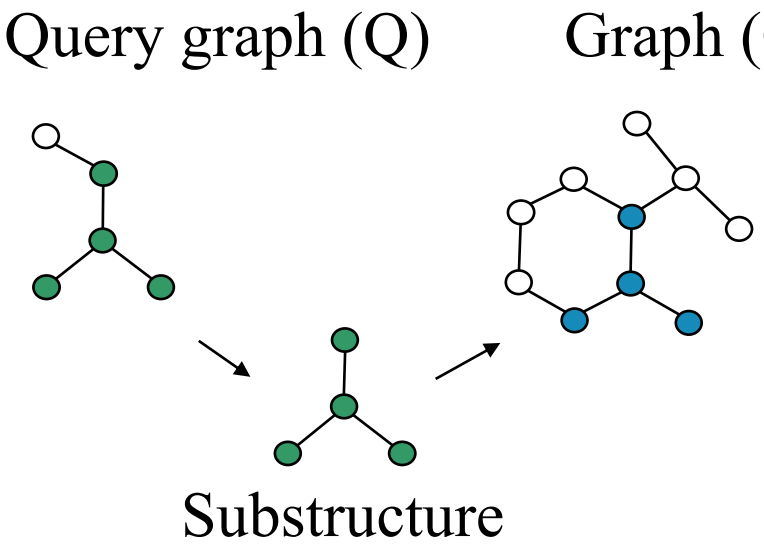
- Querying graph databases:
  - Given a graph database and a query graph, find all the graphs containing this query graph



Scalability Issue

- Naïve solution
  - Sequential scan (Disk I/O)
  - Subgraph isomorphism test (NP-complete)
- Problem: Scalability is a big issue
- An indexing mechanism is needed

Indexing Strategy



If graph G contains query graph Q, G should contain any substructure of Q

Remarks

- Index substructures of a query graph to prune graphs that do not contain these substructures

Indexing Framework

- Two steps in processing graph queries

Step 1. Index Construction

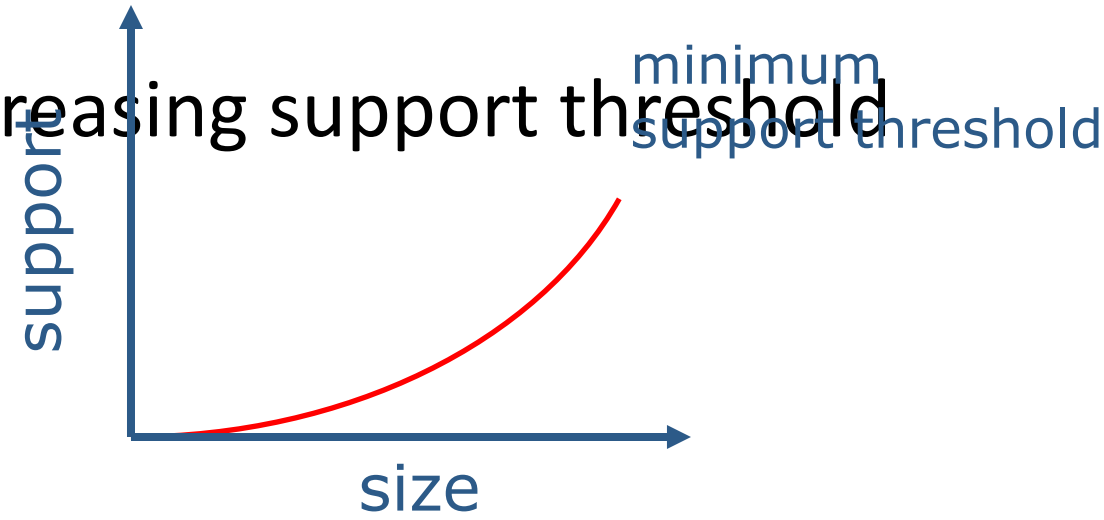
- Enumerate structures in the graph database, build an inverted index between structures and graphs

Step 2. Query Processing

- Enumerate structures in the query graph
- Calculate the candidate graphs containing these structures
- Prune the false positive answers by performing subgraph isomorphism test

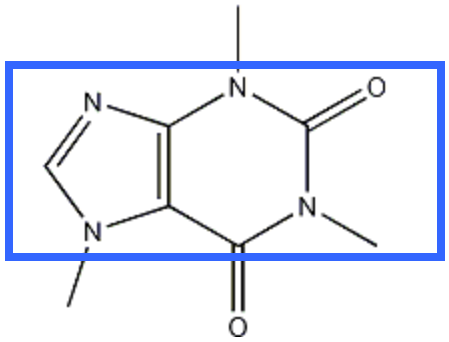
Why Frequent Structures?

- We cannot index (or even search) all of substructures
- Large structures will likely be indexed well by their substructures
- Size-increasing support threshold

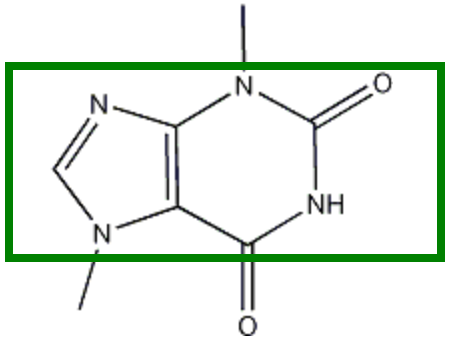


Structure Similarity Search

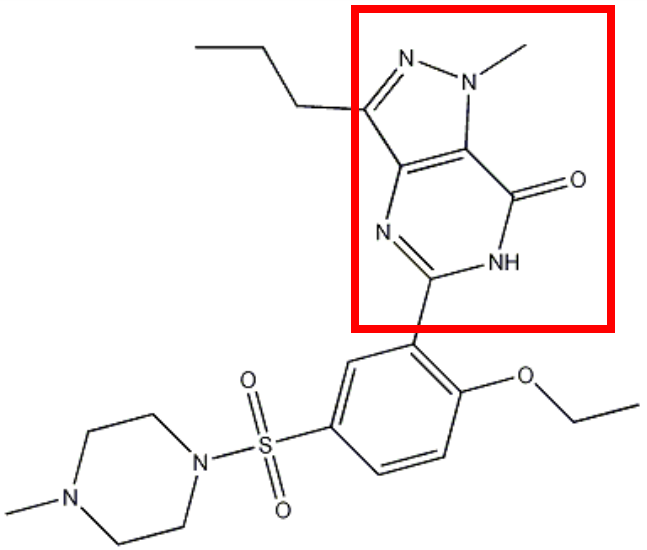
- **CHEMICAL COMPOUNDS**



(a) caffeine

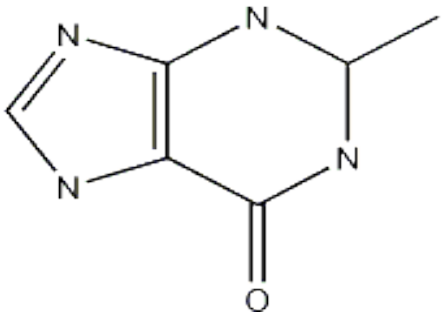


(b) diurobromine



(c) sildenafil

- **QUERY GRAPH**



Substructure Similarity Measure

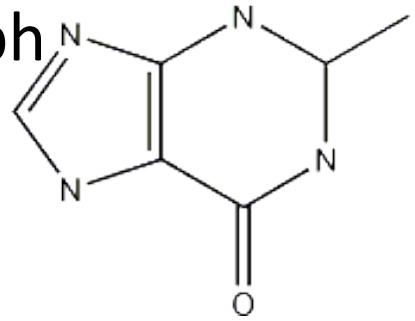
- **Feature-based similarity measure**
  - Each graph is represented as a feature vector

$$X = \{x_1, x_2, \ldots, x_n\}$$

  - Similarity is defined by the distance of their corresponding vectors
  - Advantages
    - Easy to index
    - Fast
    - Rough measure

Some "Straightforward" Methods

- Method1: Directly compute the similarity between the graphs in the DB and the query graph
  - Sequential scan
  - Subgraph similarity computation



- Method 2: Form a set of subgraph queries from the original query graph and use the exact subgraph search
  - Costly: If we allow 3 edges to be missed in a 20-edge query graph, it may generate 1,140 subgraphs

Index: Precise vs. Approximate Search

- Precise Search
  - Use frequent patterns as indexing features
  - Select features in the **database space** based on their selectivity
  - Build the index

- Approximate Search
  - Hard to build indices covering similar subgraphs
    - explosive number of subgraphs in databases
  - Idea: (1) keep the index structure

    (2) select features in the **query space**

Outline

- Graph Pattern Mining
  - Mining Frequent Subgraph Patterns
  - Graph Indexing
  - Graph Similarity Search
- Graph Classification
  - Graph pattern-based approach
  - Machine Learning approaches
- Graph Clustering
  - Link-density-based approach

Substructure-Based Graph Classification

- **Basic idea**
  - Extract graph substructures $F = \{g_1, \ldots, g_n\}$
  - Represent a graph with a feature vector $\mathbf{x} = \{x_1, \ldots, x_n\}$,
    - where x_i is the frequency of g_i in that graph
  - Build a classification model

- **Different features and representative work**
  - Fingerprint
  - Maccs keys
  - Tree and cyclic patterns [Horvath et al.]
  - Minimal contrast subgraph [Ting and Bailey]
  - Frequent subgraphs [Deshpande et al.; Liu et al.]
  - Graph fragments [Wale and Karypis]

Direct Mining of Discriminative Patterns

- **Avoid mining the whole set of patterns**
  - Harmony [Wang and Karypis]
  - DDPMine [Cheng et al.]
  - LEAP [Yan et al.]
  - MbT [Fan et al.]
- **Find the most discriminative pattern**
  - A search problem?
  - An optimization problem?
- **Extensions**
  - Mining top-k discriminative patterns
  - Mining approximate/weighted discriminative patterns

Graph Kernels

- Motivation:
  - Kernel based learning methods doesn't need to access data points
    - They rely on the kernel function between the data points
  - Can be applied to any complex structure provided you can define a kernel function on them

- Basic idea:
  - Map each graph to some significant set of patterns
  - Define a kernel on the corresponding sets of patterns

Kernel-based Classification

- Random walk
  - Basic Idea: count the matching random walks between the two graphs
  - Marginalized Kernels
    - Gärtner '02, Kashima et al. '02, Mahé et al.'04

$$K(G_1, G_2) = \sum_{h_1} \sum_{h_2} p(h_1) p(h_2) K_L(l(h_1), l(h_2))$$

- h_1 and h_2 are paths in graphs G_1 and G_2
- $p(h_1)$ and $p(h_2)$ are probability distributions on paths
- $K_L(l(h_1), l(h_2))$ is a kernel between paths, e.g.,

$$K_L(l_1, l_2) = \begin{cases} 1 & \text{if } l_1 = l_2, \\ 0 & otherwise. \end{cases}$$

Boosting in Graph Classification

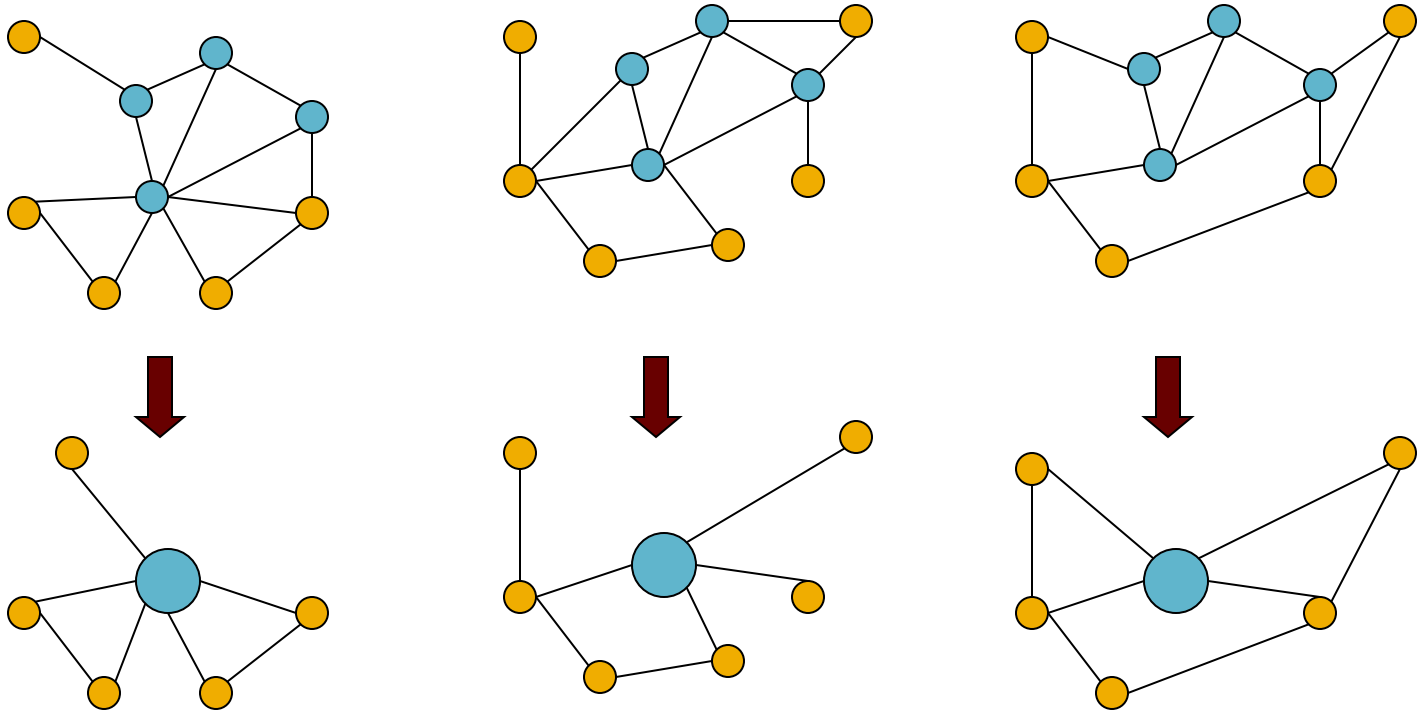
- Decision stumps
  - Simple classifiers in which the final decision is made by single features
  - A rule is a tuple $< t, y >$
    - If a molecule contains substructure t, it is classified as y.
  - Gain $$h_{<t,y>}(\mathbf{x}) = \begin{cases} y & \text{if } t \subseteq \mathbf{x}, \\ -y & otherwise. \end{cases}$$
  - Applying boo $$gain(< t, y >) = \sum_{i=1}^{n} y_i h_{<t,y>}(\mathbf{x}_i)$$ $$gain(< t, y >) = \sum_{i=1}^{n} y_i d_i h_{<t,y>}(\mathbf{x}_i)$$

Outline

- Graph Pattern Mining
  - Mining Frequent Subgraph Patterns
  - Graph Indexing
  - Graph Similarity Search
- Graph Classification
  - Graph pattern-based approach
  - Machine Learning approaches
- Graph Clustering
  - Link-density-based approach

Graph Compression

- Extract common subgraphs and simplify graphs by condensing these subgraphs into nodes

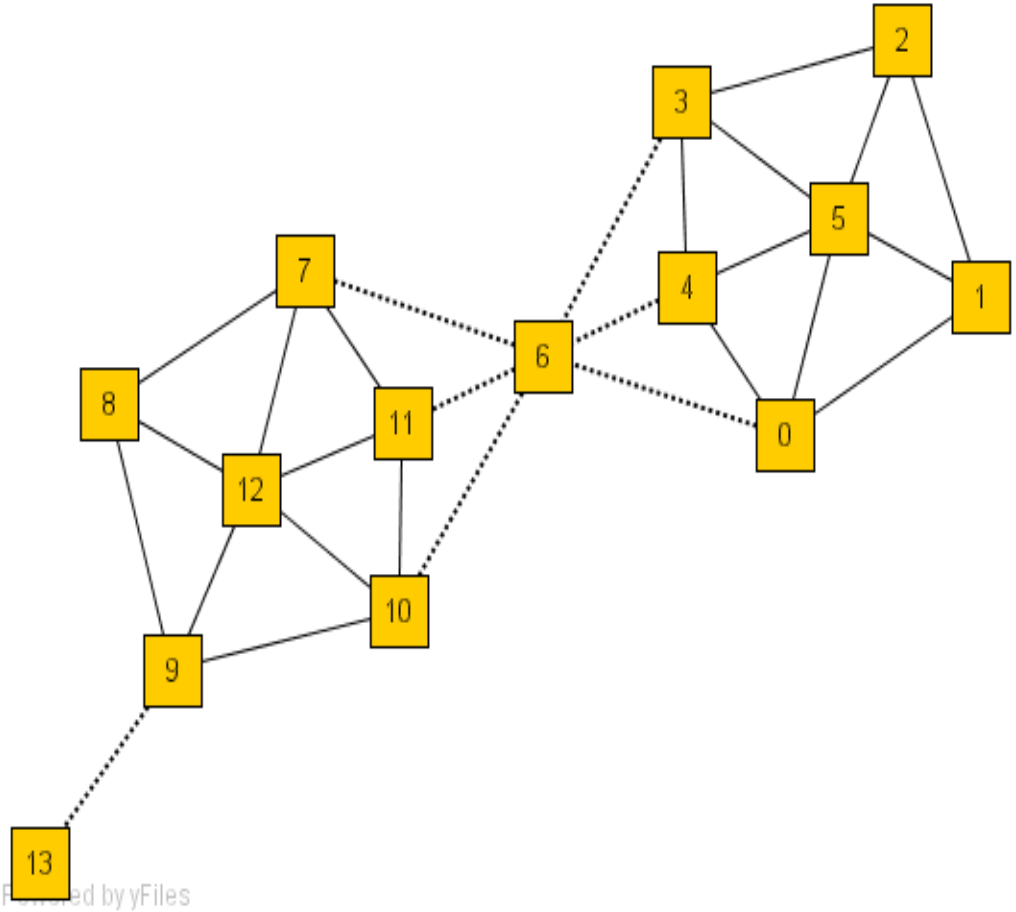


Graph/Network Clustering Problem

- Networks made up of the mutual relationships of data elements usually have an underlying structure
  - Because relationships are complex, it is difficult to discover these structures.
    - How can the structure be made clear?
- Given simple information of who associates with whom, could one identify clusters of individuals with common interests or special relationships?
  - E.g., families, cliques, terrorist cells…

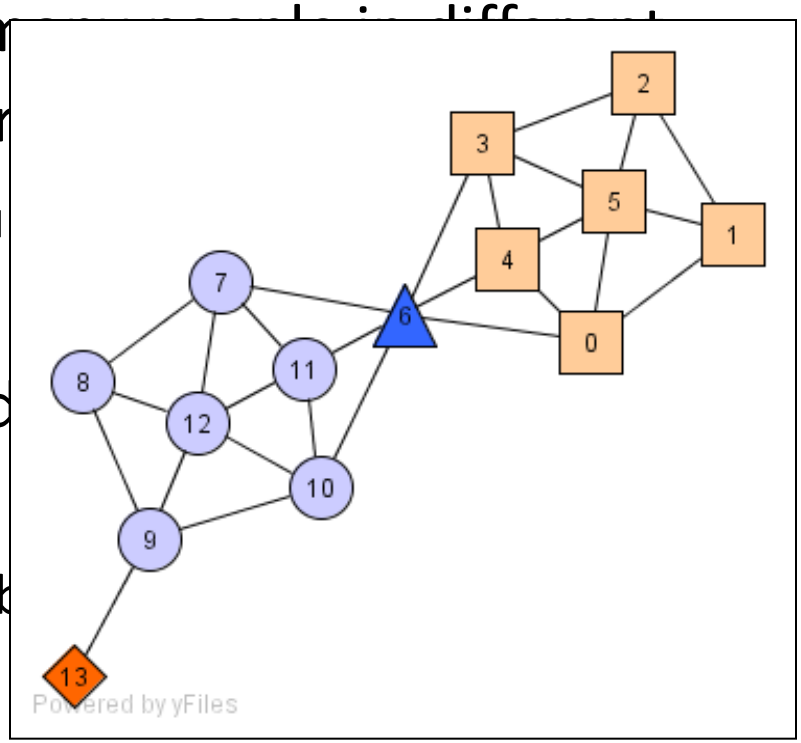
An Example of Networks

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



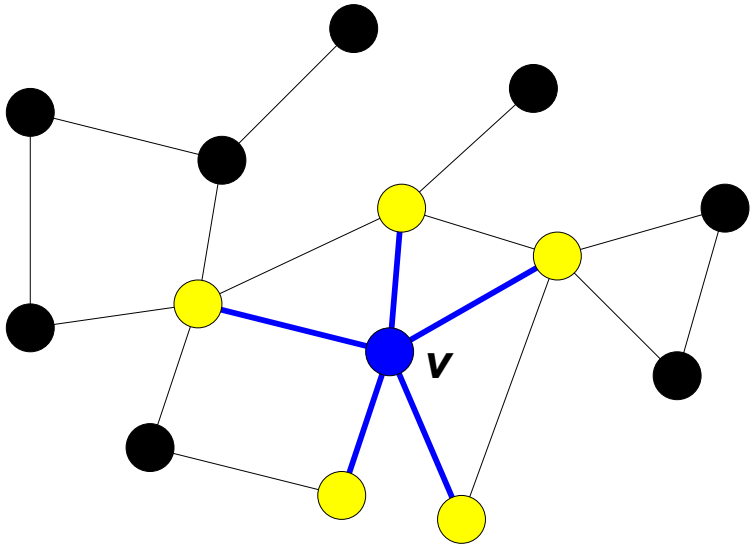
A Social Network Model

- 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 m
  groups but belong to no single gr
  - E.g., politicians bridge multiple grou
- Individuals who are outliers resic
  society
  - E.g., Hermits know few people and k



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

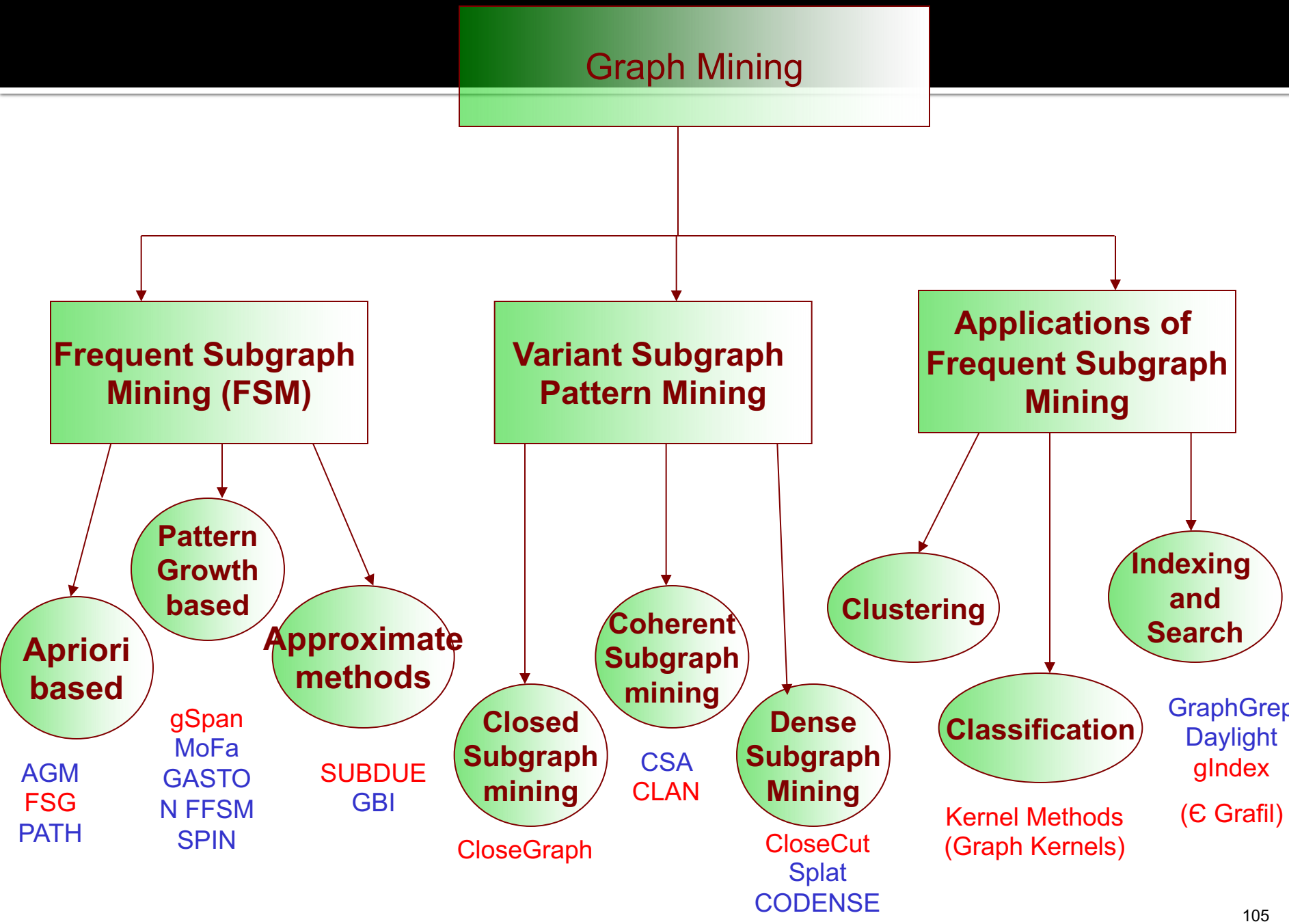


Structure Similarity

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

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

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


Graph Mining
Frequent Subgraph Mining (FSM)
Variant Subgraph Pattern Mining
Applications of Frequent Subgraph Mining
Apriori based
Pattern Growth based
Approximate methods
Closed Subgraph mining
Coherent Subgraph mining
Dense Subgraph Mining
Clustering
Classification
Indexing and Search
AGM
FSG
PATH
gSpan
MoFa
GASTO
N FFSM
SPIN
SUBDUE
GBI
CloseGraph
CSA
CLAN
CloseCut
Splat
CODENSE
Kernel Methods
(Graph Kernels)
GraphGrep
Daylight
gIndex
(Є Grafil)

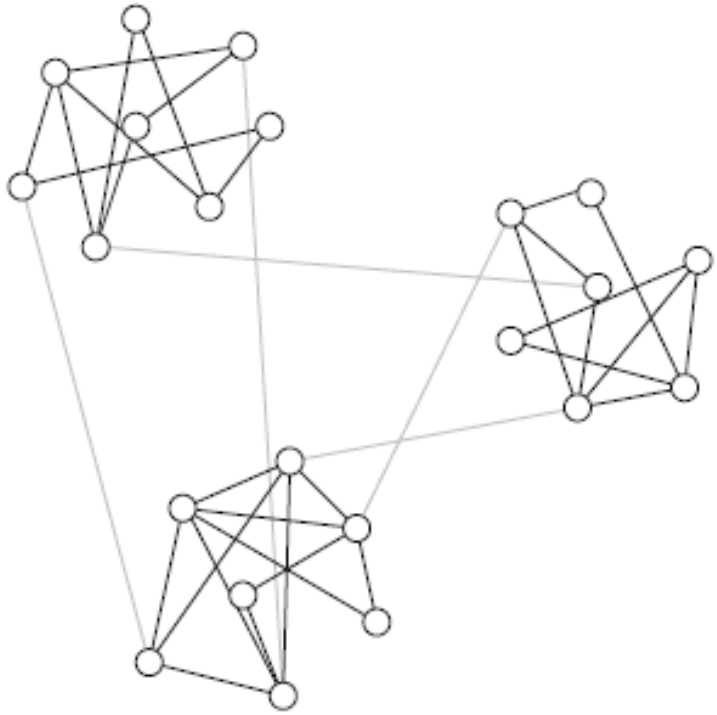


FIG. 1: A schematic representation of a network with community structure. In this network there are three communities of densely connected vertices (circles with solid lines), with a much lower density of connections (gray lines) between them.

Community

- Community: It is formed by individuals such that those within a group interact with each other more frequently than with those outside the group
  - a.k.a. group, cluster, cohesive subgroup, module in different contexts
- Community detection: discovering groups in a network where individuals' group memberships are not explicitly given

- Why communities in social media?
  - Human beings are social
  - Easy-to-use social media allows people to extend their social life in unprecedented ways
  - Difficult to meet friends in the physical world, but much easier to find friend online with similar interests
  - Interactions between nodes can help determine communities

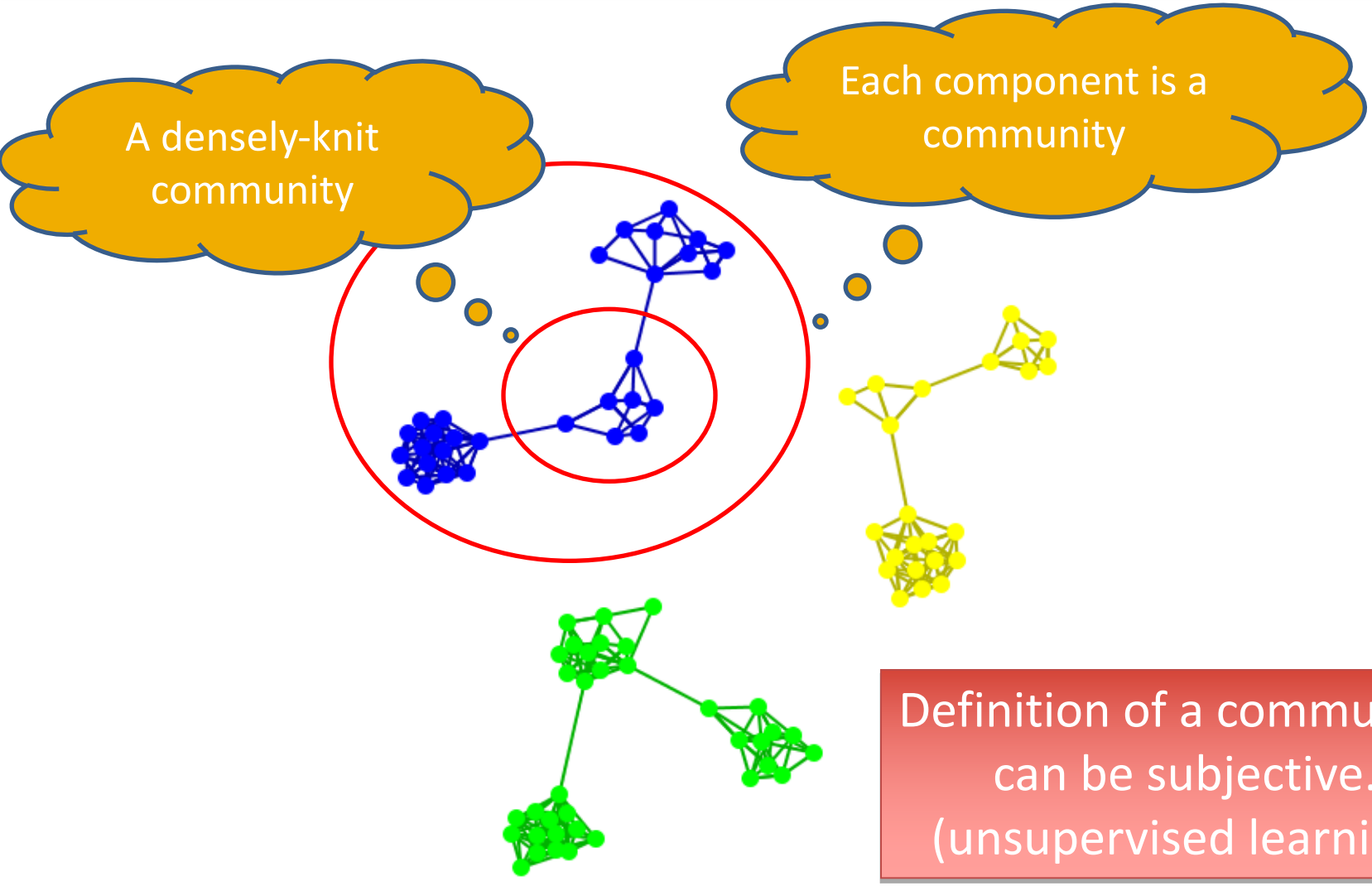
Communities in Social Media

- Two types of groups in social media
  - Explicit Groups: formed by user subscriptions
  - Implicit Groups: implicitly formed by social interactions
- Some social media sites allow people to join groups, is it necessary to extract groups based on network topology?
  - Not all sites provide community platform
  - Not all people want to make effort to join groups
  - Groups can change dynamically
- Network interaction provides rich information about the relationship between users
  - Can complement other kinds of information, e.g. user profile
  - Help network visualization and navigation
  - Provide basic information for other tasks, e.g. recommendation

Note that each of the above three points can be a research topic.

COMMUNITY DETECTION

Subjectivity of Community Definition



Definition of a community can be subjective. (unsupervised learning)

Taxonomy of Community Criteria

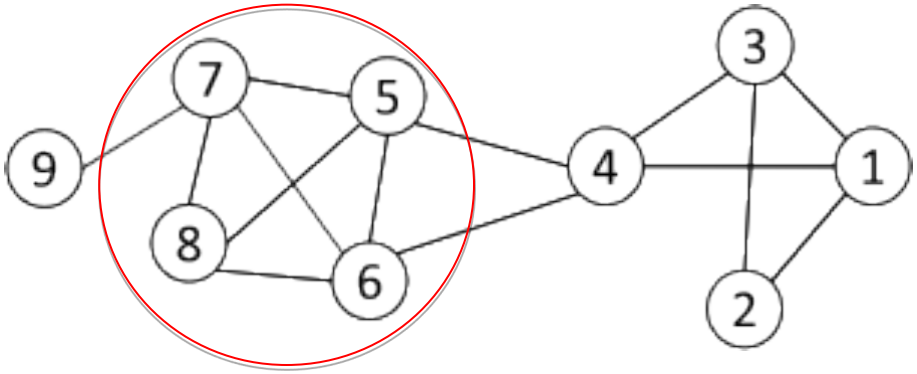
- Criteria vary depending on the tasks
- Roughly, community detection methods can be divided into 4 categories (not exclusive):
- Node-Centric Community
  - Each node in a group satisfies certain properties
- Group-Centric Community
  - Consider the connections within a group as a whole. The group has to satisfy certain properties without zooming into node-level
- **Network-Centric Community**
  - Partition the whole network into several disjoint sets
- Hierarchy-Centric Community
  - Construct a hierarchical structure of communities

Node-Centric Community Detection

- Nodes satisfy different properties
  - Complete Mutuality
    - cliques
  - Reachability of members
    - k-clique, k-clan, k-club
  - Nodal degrees
    - k-plex, k-core
  - Relative frequency of Within-Outside Ties
    - LS sets, Lambda sets
- Commonly used in traditional social network analysis
- Here, we discuss some representative ones

Complete Mutuality: Cliques

- Clique: a maximum complete subgraph in which all nodes are adjacent to each other



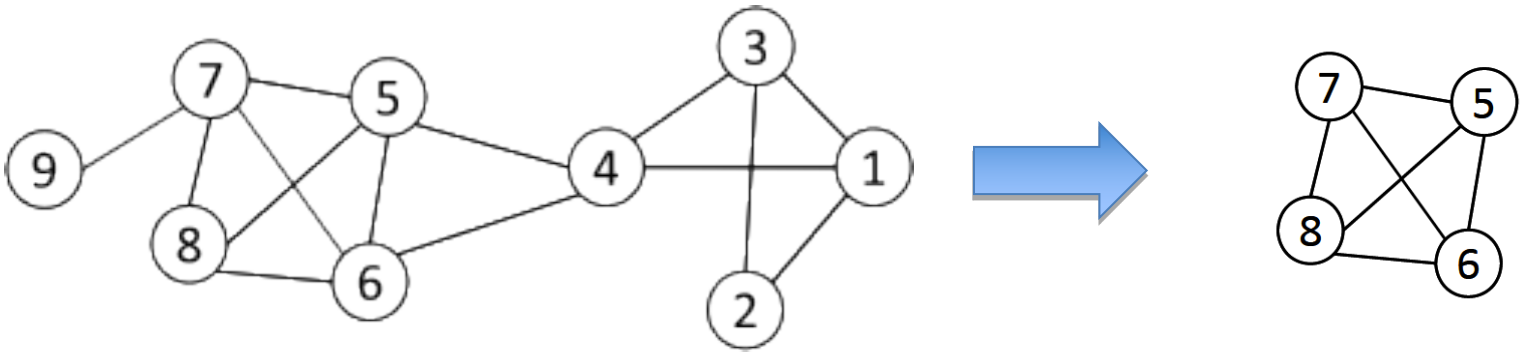
Nodes 5, 6, 7 and 8 form a clique

- NP-hard to find the maximum clique in a network
- Straightforward implementation to find cliques is very expensive in time complexity

Finding the Maximum Clique

- In a clique of size k, each node maintains degree >= k-1
  - Nodes with degree < k-1 will not be included in the maximum clique
- Recursively apply the following pruning procedure
  - Sample a sub-network from the given network, and find a clique in the sub-network, say, by a greedy approach
  - Suppose the clique above is size k, in order to find out a *larger* clique, all nodes with degree <= k-1 should be removed.
- Repeat until the network is small enough
- Many nodes will be pruned as social media networks follow a power law distribution for node degrees

Maximum Clique Example



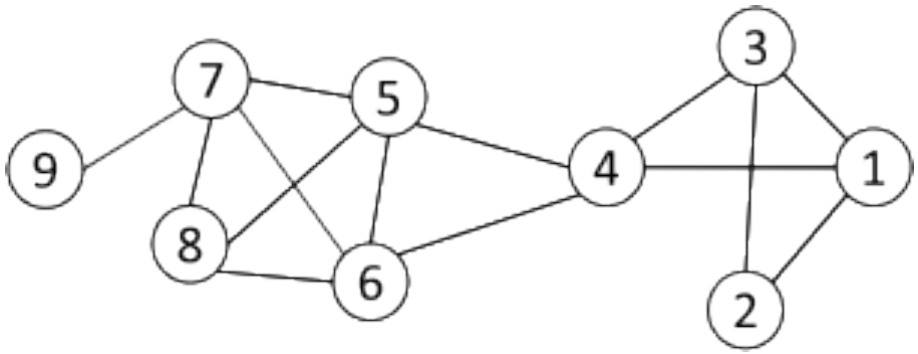
- Suppose we sample a sub-network with nodes {1-9} and find a clique {1, 2, 3} of size 3
- In order to find a clique >3, remove all nodes with degree <=3-1=2
  - Remove nodes 2 and 9
  - Remove nodes 1 and 3
  - Remove node 4

Clique Percolation Method (CPM)

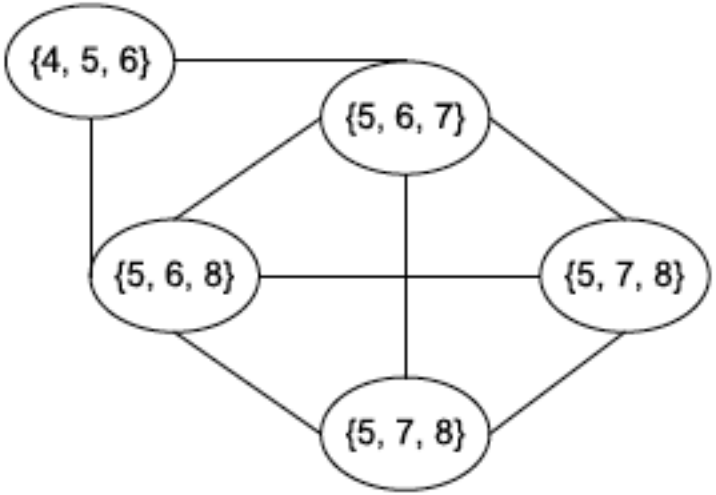
- Clique is a very strict definition, unstable
- Normally use cliques as a core or a seed to find larger communities

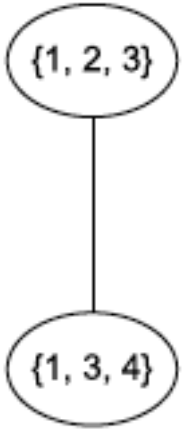
- CPM is such a method to find overlapping communities
  - **Input**
    - A parameter k, and a network
  - **Procedure**
    - Find out all cliques of size k in a given network
    - Construct a clique graph. Two cliques are adjacent if they share k-1 nodes
    - Each connected component in the clique graph forms a community

CPM Example



Cliques of size 3:
{1, 2, 3}, {1, 3, 4}, {4, 5, 6}, {5, 6, 7}, {5, 6, 8}, {5, 7, 8}, {6, 7, 8}

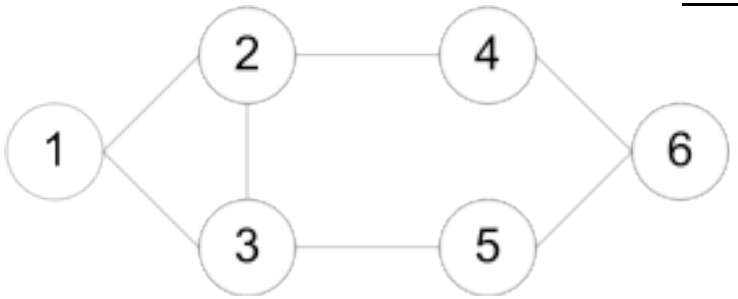




Communities:
{1, 2, 3, 4}
{4, 5, 6, 7, 8}

Reachability : k-clique, k-club

- Any node in a group should be reachable in k hops
- k-clique: a maximal subgraph in which the largest geodesic distance between any two nodes <= k
- k-club: a substructure of diameter <= k



Cliques: {1, 2, 3}
2-cliques: {1, 2, 3, 4, 5}, {2, 3, 4, 5, 6}
2-clubs: {1,2,3,4}, {1, 2, 3, 5}, {2, 3, 4, 5, 6}

- A k-clique might have diameter larger than k in the subgraph
  - E.g. {1, 2, 3, 4, 5}
- Commonly used in traditional SNA
- Often involves combinatorial optimization

Group-Centric Community Detection: Density-Based Groups

- The group-centric criterion requires the whole group to satisfy a certain condition
  - E.g., the group density >= a given threshold
- A subgraph $G_s(V_s, E_s)$ is a $\gamma - dense$ quasi-clique if

$$\frac{2|E_s|}{|V_s|(|V_s|-1)} \geq \gamma$$

where the denominator is the maximum number of degrees.

- A similar strategy to that of cliques can be used
  - Sample a subgraph, and find a maximal $\gamma - dense$ quasi-clique (say, of size $|V_s|$)
  - Remove nodes with degree less than the average degree

$$< \quad |V_s|\gamma \leq \frac{2|E_s|}{|V_s|-1}$$

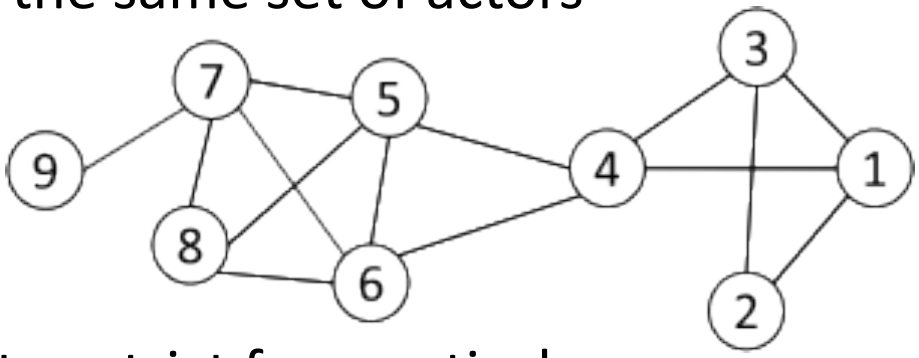
Network-Centric Community Detection

- Network-centric criterion needs to consider the connections within a network globally
- Goal: partition nodes of a network into disjoint sets
- Approaches:
  - **(1) Clustering based on vertex similarity**
  - (2) Latent space models (multi-dimensional scaling)
  - (3) Block model approximation
  - (4) Spectral clustering
  - **(5) Modularity maximization**

Clustering based on Vertex Similarity

- Apply k-means or similarity-based clustering to nodes
- Vertex similarity is defined in terms of the similarity of their neighborhood
- Structural equivalence: two nodes are structurally equivalent iff they are connecting to the same set of actors

Nodes 1 and 3 are structurally equivalent;
So are nodes 5 and 6.



- Structural equivalence is too strict for practical use.

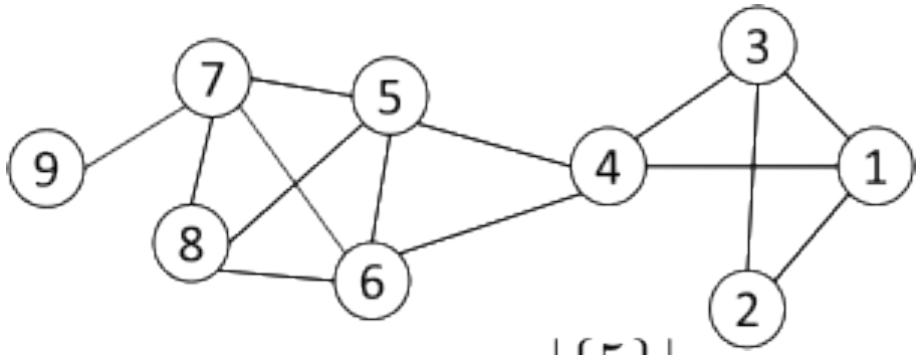
Vertex Similarity

- Jaccard Similarity

$$Jaccard(v_i, v_j) = \frac{|N_i \cap N_j|}{|N_i \cup N_j|}$$

- Cosine similarity

$$Cosine(v_i, v_j) = \frac{|N_i \cap N_j|}{\sqrt{|N_i| \cdot |N_j|}}$$

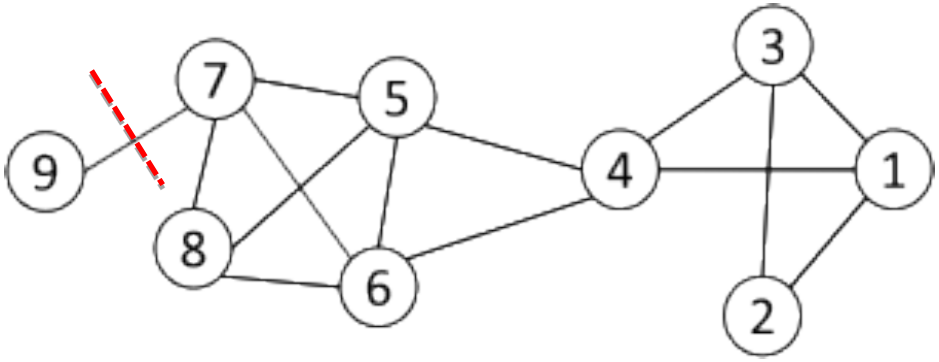


$$Jaccard(4,6) = \frac{|\{5\}|}{|\{1,3,4,5,6,7,8\}|} = \frac{1}{7}$$

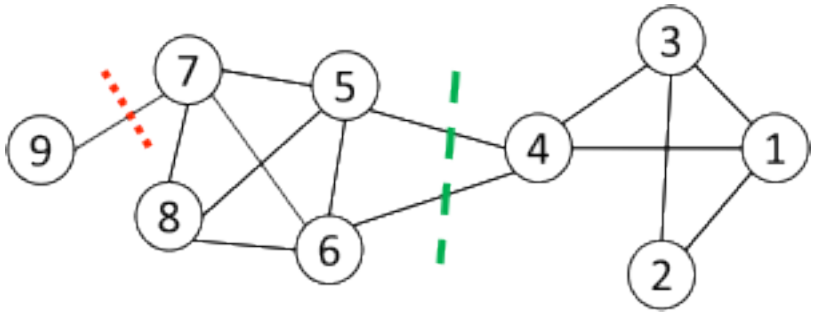
$$cosine(4,6) = \frac{1}{\sqrt{4 \cdot 4}} = \frac{1}{4}$$

Cut

- Most interactions are within group whereas interactions between groups are few
- community detection → minimum cut problem
- Cut: A partition of vertices of a graph into two disjoint sets
- Minimum cut problem: find a graph partition such that the number of edges between the two sets is minimized



Ratio Cut & Normalized Cut



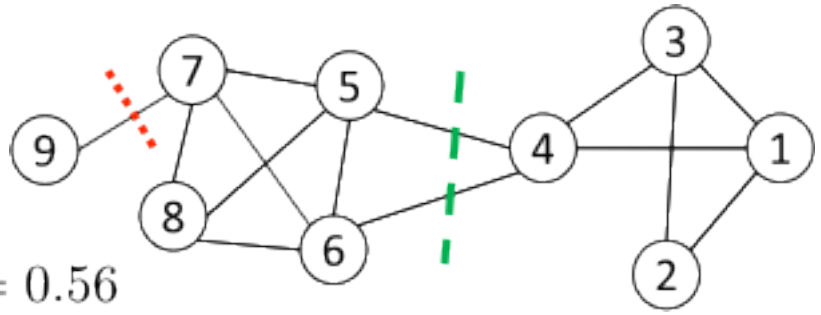
- Minimum cut often returns an imbalanced partition, with one set being a singleton, e.g. node 9
- Change the objective function to consider community size

$$\text{Ratio Cut}(\pi) = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(C_i, \bar{C}_i)}{|C_i|},$$

$$\text{Normalized Cut}(\pi) = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(C_i, \bar{C}_i)}{vol(C_i)}$$

C_i: a community
$|C_i|$: number of nodes in C_i
$vol(C_i)$: sum of degrees in C_i

Ratio Cut & Normalized Cut Example



For partition in red: π_1

$$\text{Ratio Cut}(\pi_1) = \frac{1}{2}\left(\frac{1}{1} + \frac{1}{8}\right) = 9/16 = 0.56$$

$$\text{Normalized Cut}(\pi_1) = \frac{1}{2}\left(\frac{1}{1} + \frac{1}{27}\right) = 14/27 = 0.52$$

For partition in green: π_2

$$\text{Ratio Cut}(\pi_2) = \frac{1}{2}\left(\frac{2}{4} + \frac{2}{5}\right) = 9/20 = 0.45 < \text{Ratio Cut}(\pi_1)$$

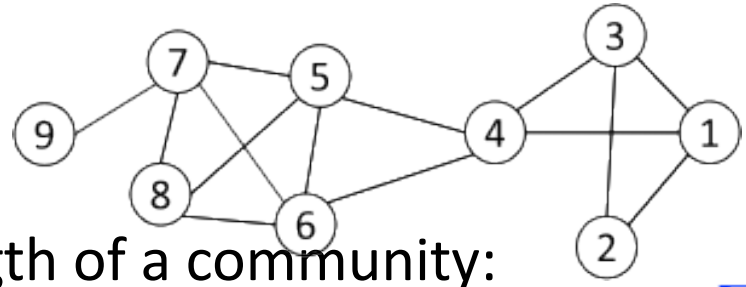
$$\text{Normalized Cut}(\pi_2) = \frac{1}{2}\left(\frac{2}{12} + \frac{2}{16}\right) = 7/48 = 0.15 < \text{Normalized Cut}(\pi_1)$$

Both ratio cut and normalized cut prefer a balanced partition

Modularity Maximization

Modularity measures the strength of a community partition by taking into account the degree distribution

Given a network with *m* edges, the expected number of edges between two nodes with degrees d_i and d_j is $d_i d_j / 2m$



The expected number of edges between nodes 1 and 2 is 3*2/ (2*14) = 3/14

Strength of a community:

$$\sum_{i \in C, j \in C} A_{ij} - d_i d_j / 2m$$

Given the degree distribution

A larger value indicates a good community structure

$$q = \frac{1}{2m} \sum_{\ell=1}^{k} \sum_{i \in C_\ell, j \in C_\ell} (A_{ij} - d_i d_j / 2m)$$

Hierarchy-Centric Community Detection

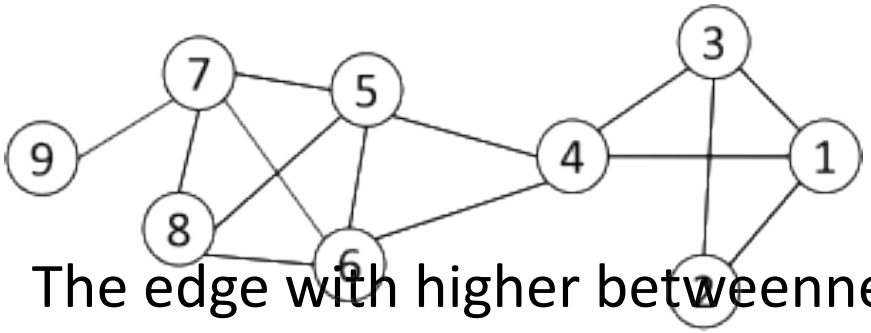
- Goal: build a hierarchical structure of communities based on network topology
- Allow the analysis of a network at different resolutions
- Representative approaches:
  - Divisive Hierarchical Clustering (top-down)
  - Agglomerative Hierarchical clustering (bottom-up)

Divisive Hierarchical Clustering

- Divisive clustering
  - Partition nodes into several sets
  - Each set is further divided into smaller ones
  - Network-centric partition can be applied for the partition
- One particular example: recursively remove the "weakest" tie
  - Find the edge with the least strength
  - Remove the edge and update the corresponding strength of each edge
- Recursively apply the above two steps until a network is decomposed into desired number of connected components.
- Each component forms a community

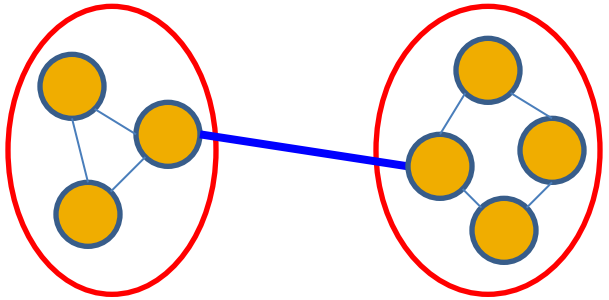
Edge Betweenness

- The strength of a tie can be measured by edge betweenness
- Edge betweenness: the number of shortest paths that pass along with the edge

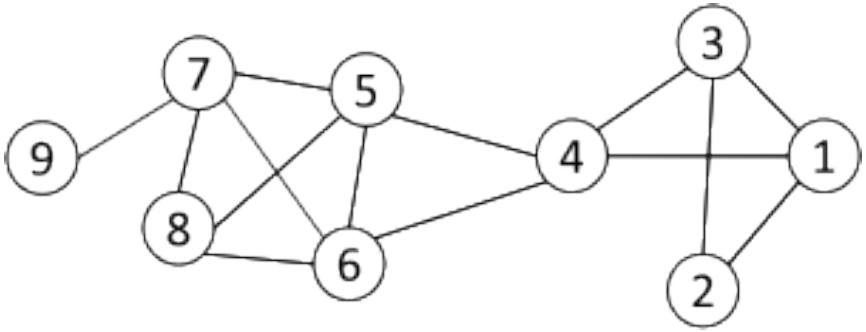


The edge betweenness of e(1, 2) is 4 (=6/2 + 1), as all the shortest paths from 2 to {4, 5, 6, 7, 8, 9} have to either pass e(1, 2) or e(2, 3), and e(1,2) is the shortest path between 1 and 2

- The edge with higher betweenness tends to be the bridge between two communities.



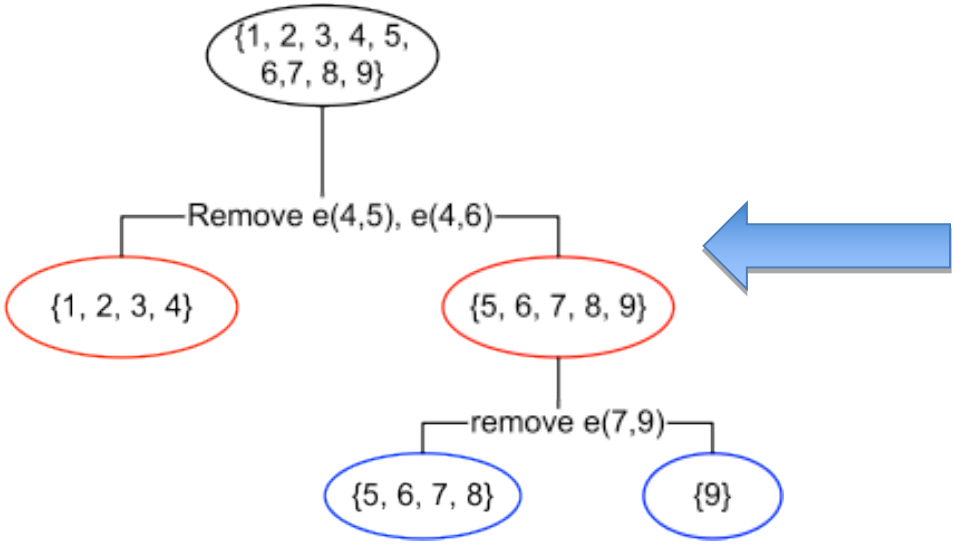
Divisive clustering based on edge betweenness



Initial betweenness value

Table 3.3: Edge Betweenness

	1	2	3	4	5	6	7	8	9
1	0	4	1	9	0	0	0	0	0
2	4	0	4	0	0	0	0	0	0
3	1	4	0	9	0	0	0	0	0
4	9	0	9	0	10	10	0	0	0
5	0	0	0	10	0	1	6	3	0
6	0	0	0	10	1	0	6	3	0
7	0	0	0	0	6	6	0	2	8
8	0	0	0	0	3	3	2	0	0
9	0	0	0	0	0	0	8	0	0



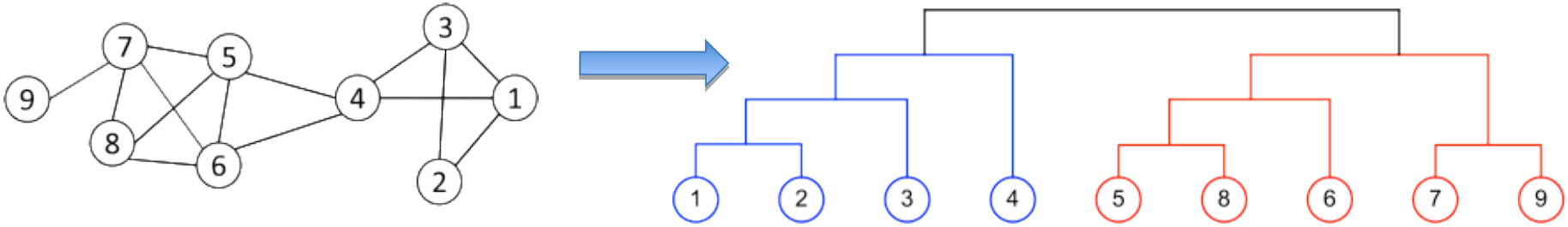
After remove e(4,5), the betweenness of e(4, 6) becomes 20, which is the highest;

After remove e(4,6), the edge e(7,9) has the highest betweenness value 4, and should be removed.

Idea: progressively removing edges with the highest betweenness

Agglomerative Hierarchical Clustering

- Initialize each node as a community
- Merge communities successively into larger communities following a certain criterion
  - E.g., based on modularity increase



Dendrogram according to Agglomerative Clustering based on Modularity

Summary of Community Detection

- Node-Centric Community Detection
  - *cliques, k-cliques, k-clubs*
- Group-Centric Community Detection
  - *quasi-cliques*
- Network-Centric Community Detection
  - *Clustering based on vertex similarity*
  - *Latent space models, block models, spectral clustering, modularity maximization*
- Hierarchy-Centric Community Detection
  - *Divisive clustering*
  - *Agglomerative clustering*

Outline

- Introduction
- Motivation and applications of Graph Mining
- Mining Frequent Subgraphs – Transaction setting
  - BFS/Apriori Approach (FSG and others)
  - DFS Approach (gSpan and others)
  - Greedy Approach
- Mining Frequent Subgraphs – Single graph setting
  - The support issue
  - The path-based algorithm

What is Data Mining?

Data Mining also known as *Knowledge Discovery in Databases* (KDD) is the process of extracting useful hidden information from very large databases in an unsupervised manner.

Mining Frequent Patterns: What is it good for?

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we classify web documents using frequent patterns?

The Apriori principle: Downward closure Property

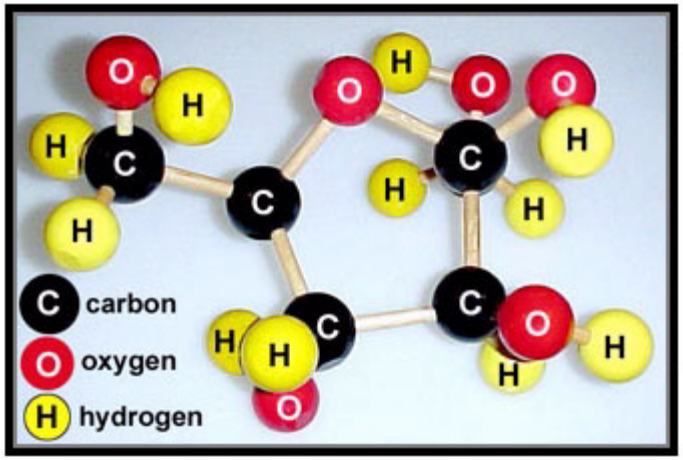
- All subsets of a frequent itemset must also be frequent
  - Because any transaction that contains *X* must also contains subset of *X*.
- If we have already verified that *X* is infrequent, there is no need to count *X* supersets because they must be infrequent too.

Outline

- Introduction
- Motivation and applications of Graph Mining
- Mining Frequent Subgraphs – Transaction setting
  - BFS/Apriori Approach (FSG and others)
  - DFS Approach (gSpan and others)
  - Greedy Approach
- Mining Frequent Subgraphs – Single graph setting
  - The support issue
  - Path mining algorithm

What Graphs are good for?

- Most of existing data mining algorithms are based on transaction representation, i.e., sets of items.
- Datasets with structures, layers, hierarchy and/or geometry often do not fit well in this transaction setting. For e.g.
  - Numerical simulations
  - 3D protein structures
  - Chemical Compounds
  - Generic XML files.



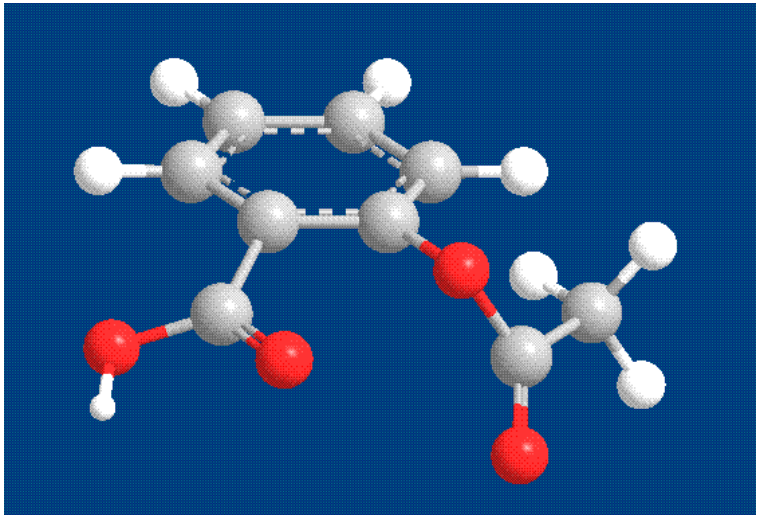
Graph Based Data Mining

- Graph Mining is the problem of discovering repetitive subgraphs occurring in the input graphs.
- Motivation:
  - finding subgraphs capable of compressing the data by abstracting instances of the substructures.
  - identifying conceptually interesting patterns

Why Graph Mining?

- Graphs are everywhere
  - 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 (NP complete or even P-SPACE !)

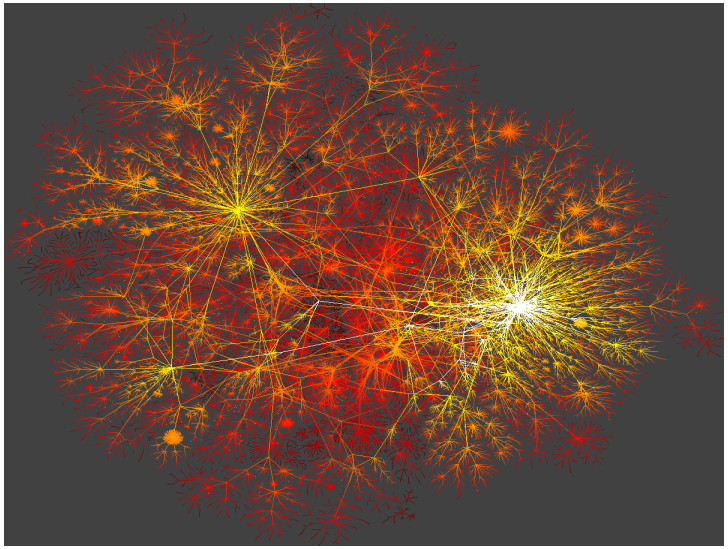
Graphs, Graphs, Everywhere



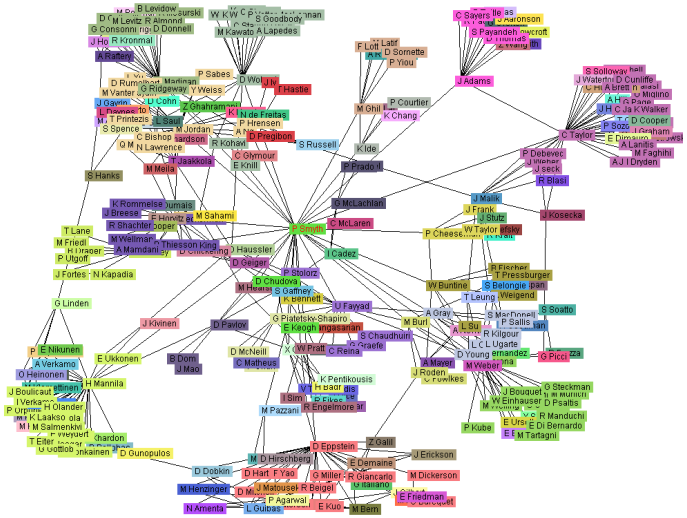
Aspirin

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

Yeast protein interaction network

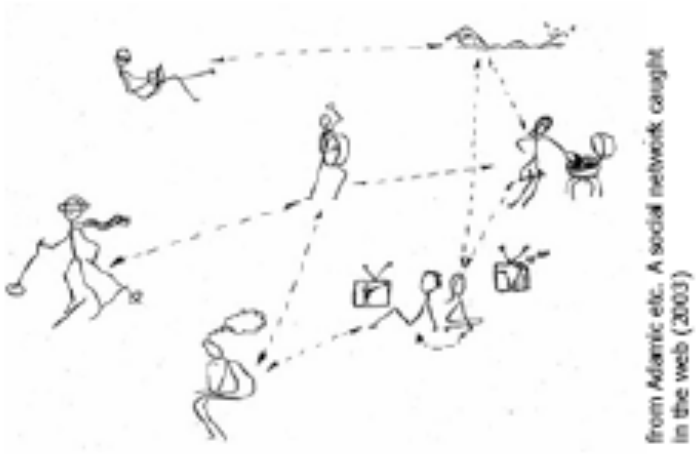


Internet

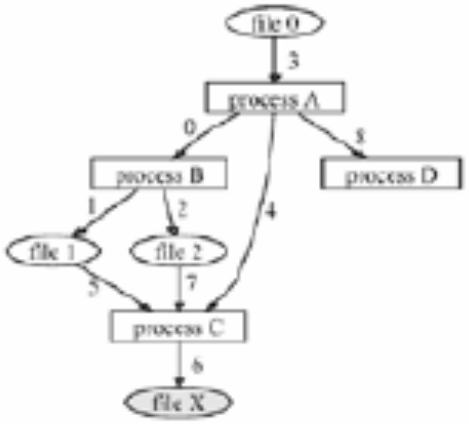


Co-author network

Graph, Graph, Everywhere (cont.)

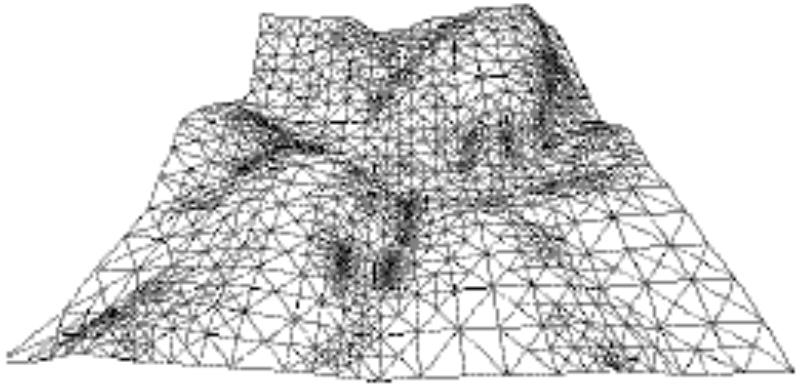


Social Network



Event Log Graph

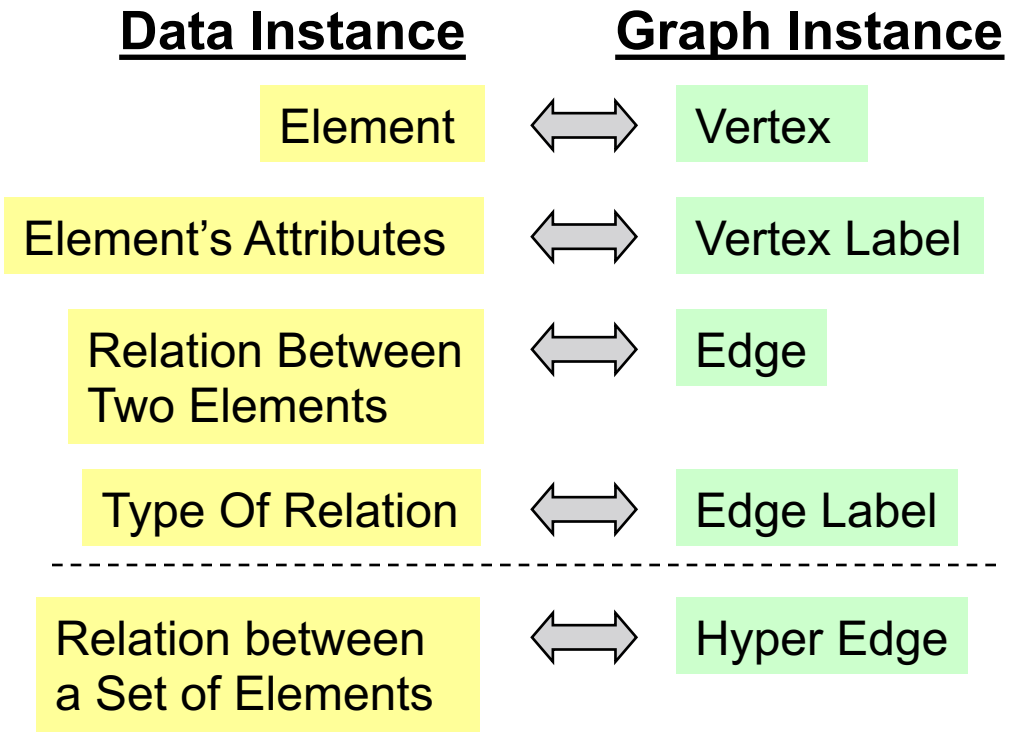
Workflow



Mesh

Modeling Data With Graphs…
Going Beyond Transactions

Graphs are suitable for capturing arbitrary relations between the various elements.



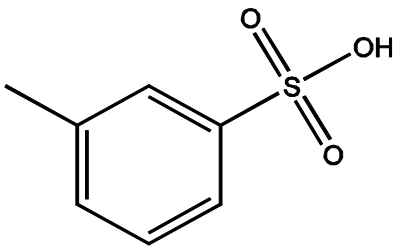
Provide enormous flexibility for modeling the underlying data as they allow the modeler to decide on what the elements should be and the type of relations to be modeled

Graph Pattern Mining

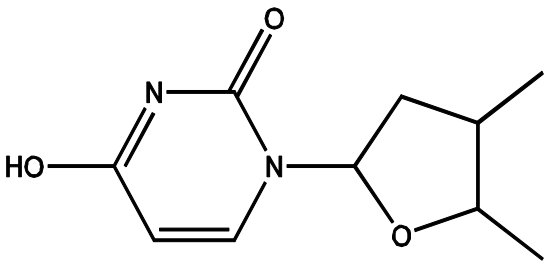
- Frequent subgraphs
  - A (sub)graph is ***frequent*** if its support (occurrence frequency) in a given dataset is no less than a minimum support threshold
- Applications of graph pattern mining:
  - Mining biochemical structures
  - Program control flow analysis
  - Mining XML structures or Web communities
  - Building blocks for graph classification, clustering, compression, comparison, and correlation analysis

Example 1

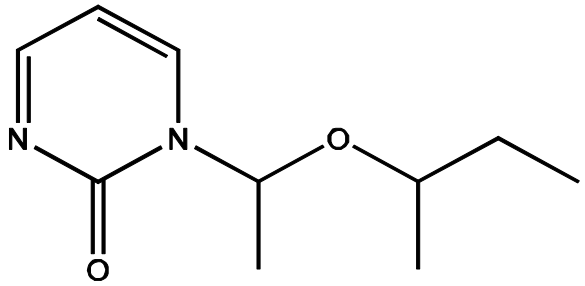
GRAPH DATASET



(T1)



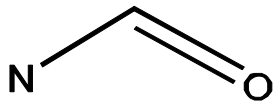
(T2)



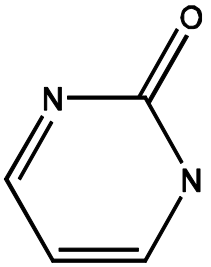
(T3)

FREQUENT PATTERNS
(MIN SUPPORT IS 2)

(1)

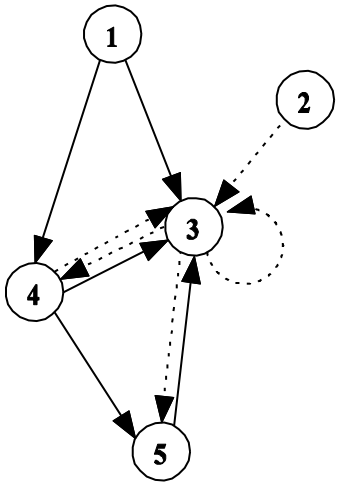


(2)

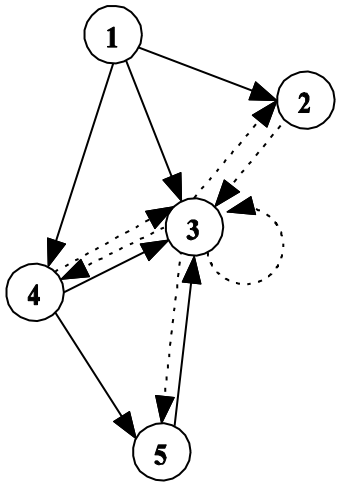


Example 2

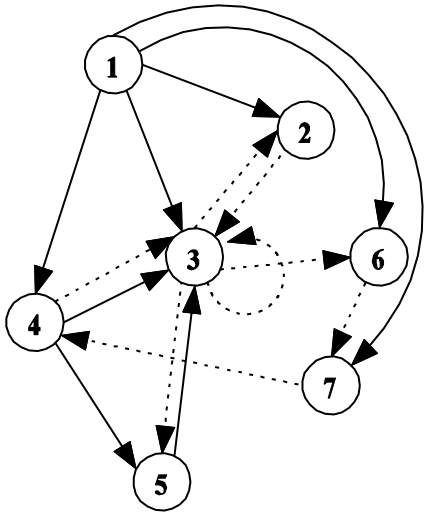
GRAPH DATASET



(1)



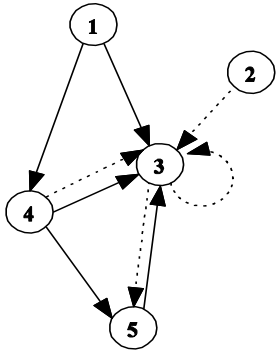
(2)



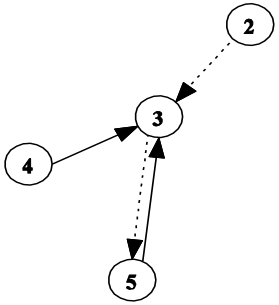
(3)

1: makepat
2: esc
3: addstr
4: getccl
5: dodash
6: in_set_2
7: stclose

FREQUENT PATTERNS (MIN SUPPORT IS 2)



(1)



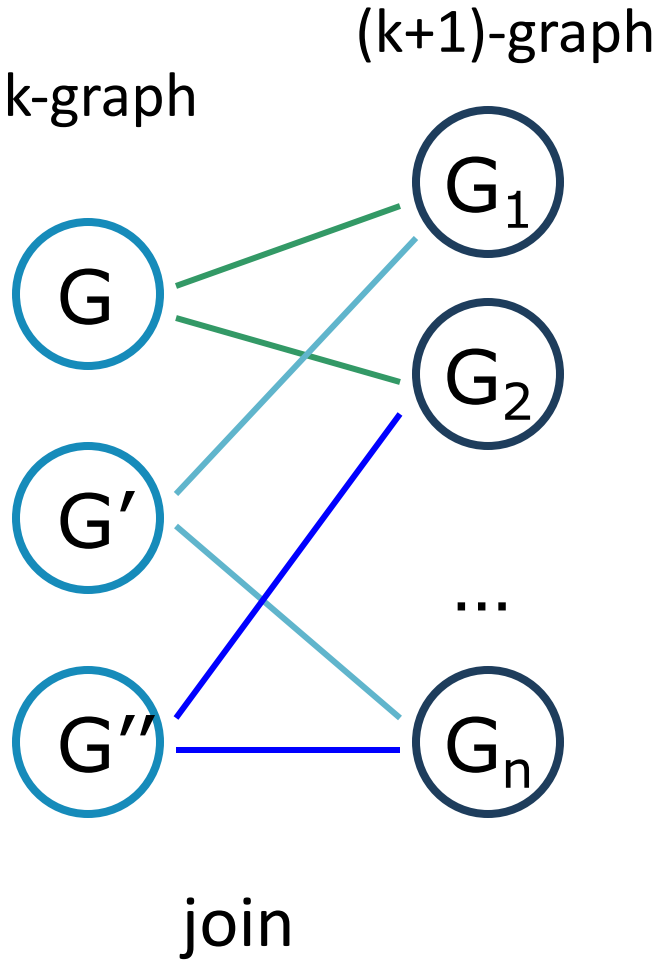
(2)

- Simple path patterns (Chen,Park,Yu 98)
- Generalized path patterns (Nanopoulos,Manolopoulos 01)
- Simple tree patterns (Lin,Liu,Zhang, Zhou 98)
- Tree-like patterns (Wang,Huiqing,Liu 98)
- General graph patterns (Kuramochi,Karypis 01, Han 02 etc.)

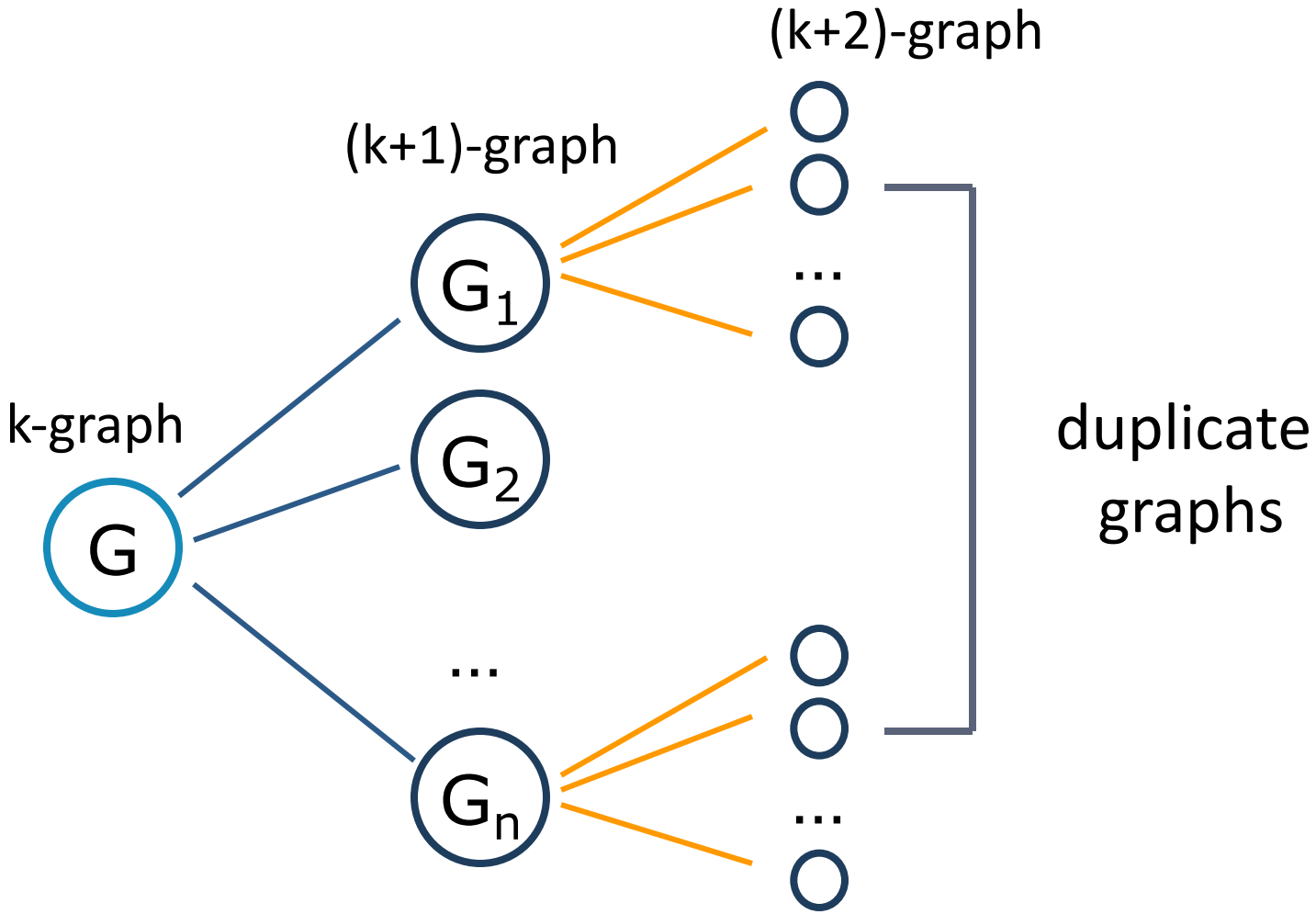
Graph mining methods

- Apriori-based approach
- Pattern-growth approach

Apriori-Based Approach



Pattern Growth Method



Outline

- Introduction
- Motivation and applications of Graph Mining
- Mining Frequent Subgraphs – Transaction setting
  - BFS/Apriori Approach (FSG and others)
  - DFS Approach (gSpan and others)
  - Greedy Approach
- Mining Frequent Subgraphs – Single graph setting
  - The support issue
  - Path mining algorithm
  - Constraint-based mining

Transaction Setting

- **Input:** (D, minSup)
  - Set of labeled-graphs transactions $D=\{T_1, T_2, \ldots, T_N\}$
  - Minimum support minSup
- **Output:** (All frequent subgraphs).
  - A subgraph is frequent if it is a subgraph of at least minSup·|D| (or #minSup) different transactions in D.
  - Each subgraph is connected.

Single graph setting

- **Input*:*** (*D*, minSup)
  - A single graph D (e.g. the Web or DBLP or an XML file)
  - Minimum support minSup
- **Output*:*** (All frequent subgraphs).
  - A subgraph is frequent if the number of its occurrences in D is above an admissible support measure (measure that satisfies the downward closure property).

Graph Mining: Transaction Setting

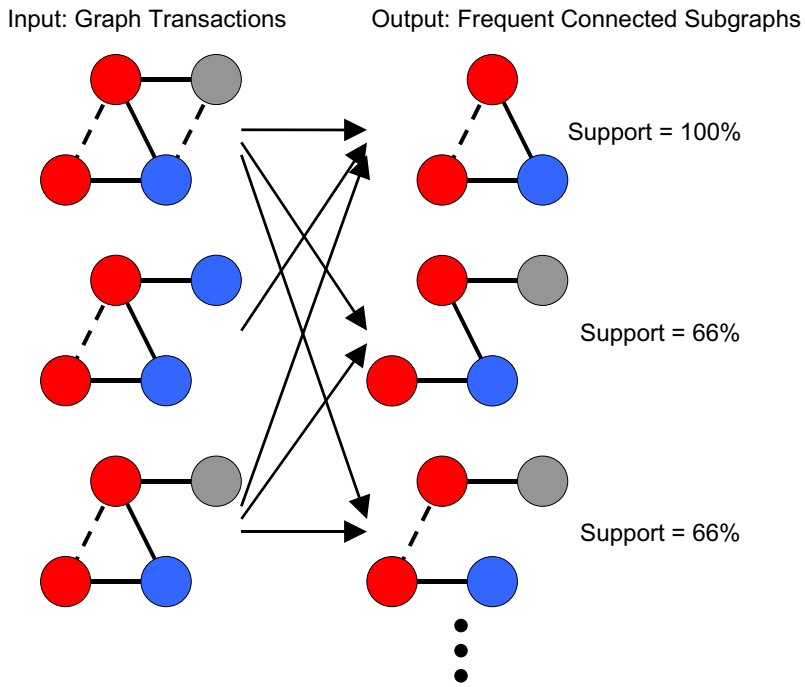
Finding Frequent Subgraphs: Input and Output

Input

- Database of graph transactions.
- Undirected simple graph (no loops, no multiples edges).
- Each graph transaction has labels associated with its vertices and edges.
- Transactions may not be connected.
- Minimum support threshold σ.

Output

- Frequent subgraphs that satisfy the minimum support constraint.
- Each frequent subgraph is connected.



Different Approaches for GM

- Apriori Approach
  - FSG
  - Path Based
- DFS Approach
  - gSpan
- Greedy Approach
  - Subdue

FSG Algorithm

[M. Kuramochi and G. Karypis. Frequent subgraph discovery. ICDM 2001]

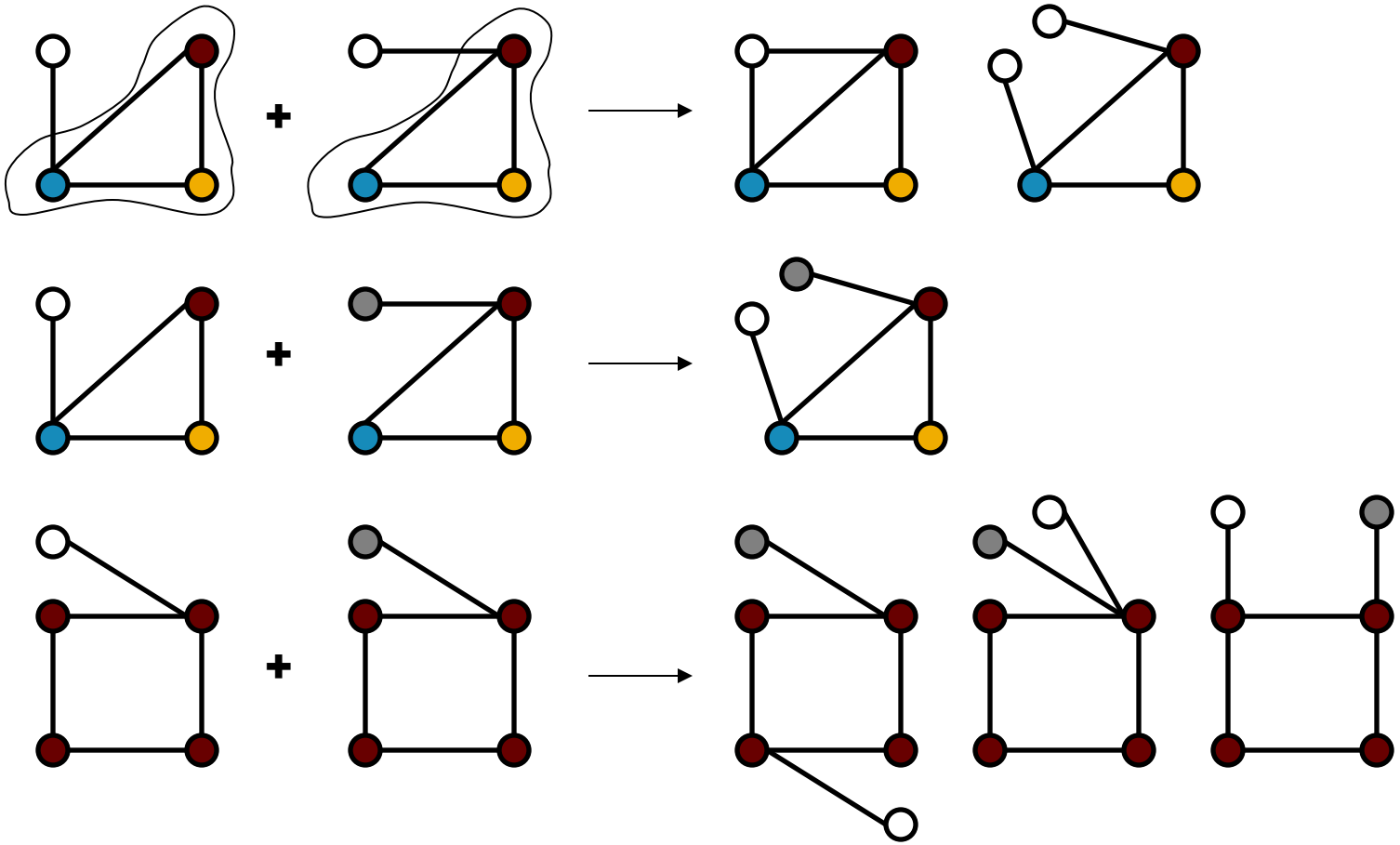
Notation: k-subgraph is a subgraph with k edges.
Init: Scan the transactions to find $\mathcal{F}_1$, the set of all frequent 1-subgraphs and 2-subgraphs, together with their counts;
For (k=3; $\mathcal{F}_{k-1} \neq \emptyset$; k++)

1. **Candidate Generation** - C_k, the set of candidate k-subgraphs, from $\mathcal{F}_{k-1}$, the set of frequent (k-1)-subgraphs;
2. **Candidates pruning** - a necessary condition of candidate to be frequent is that each of its (k-1)-subgraphs is frequent.
3. **Frequency counting** - Scan the transactions to count the occurrences of subgraphs in C_k;
4. $\mathcal{F}_k$ = { $c \in C_K$ | c has counts no less than *#minSup* }
5. Return $\mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots\ldots \cup \mathcal{F}_k$ (= $\mathcal{F}$)

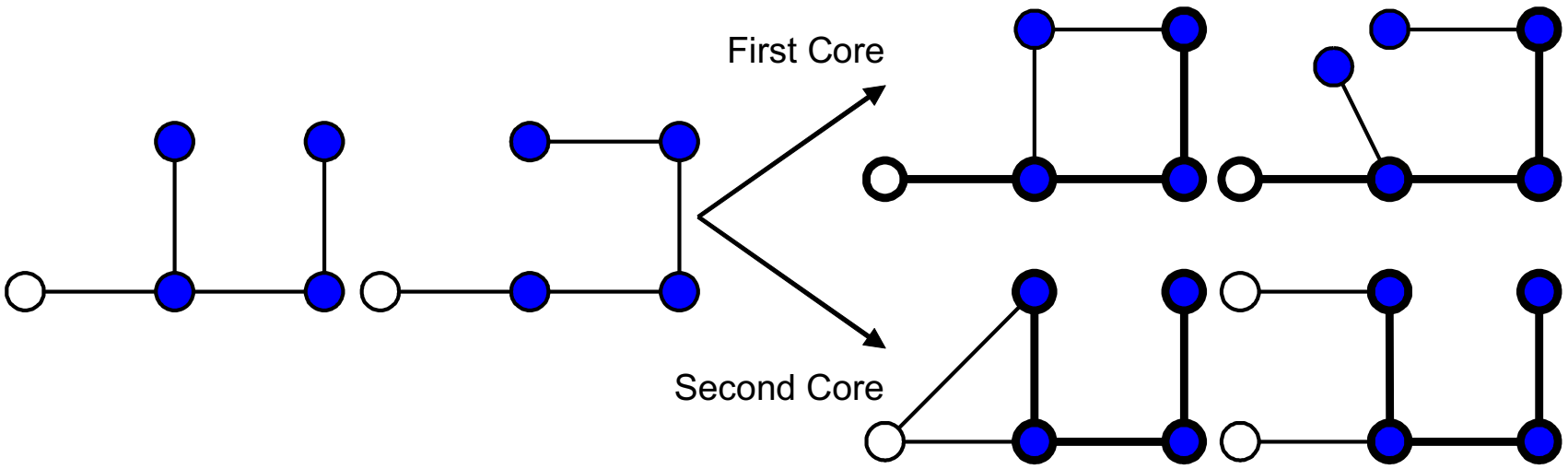
Trivial operations are complicated with graphs

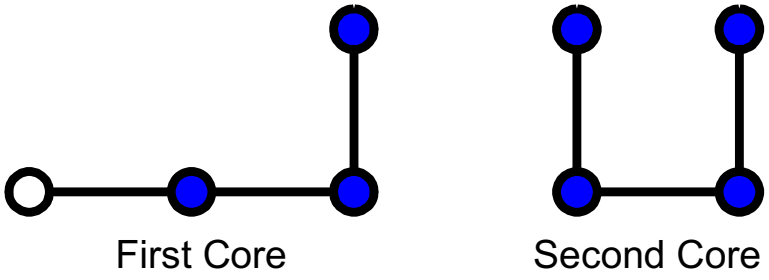
- Candidate generation
  - To determine two candidates for joining, we need to check for graph isomorphism.
- Candidate pruning
  - To check downward closure property, we need graph isomorphism.
- Frequency counting
  - Subgraph isomorphism for checking containment of a frequent subgraph.

Candidates generation (join) based on core detection



Candidate Generation Based On Core Detection (cont.)

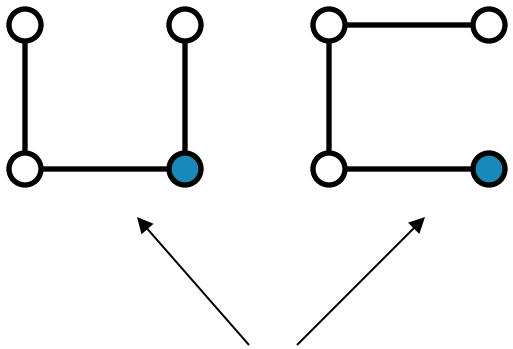




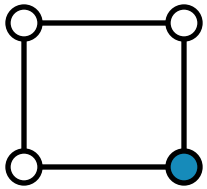
Multiple cores between two (k-1)-subgraphs

Candidate pruning:downward closure property

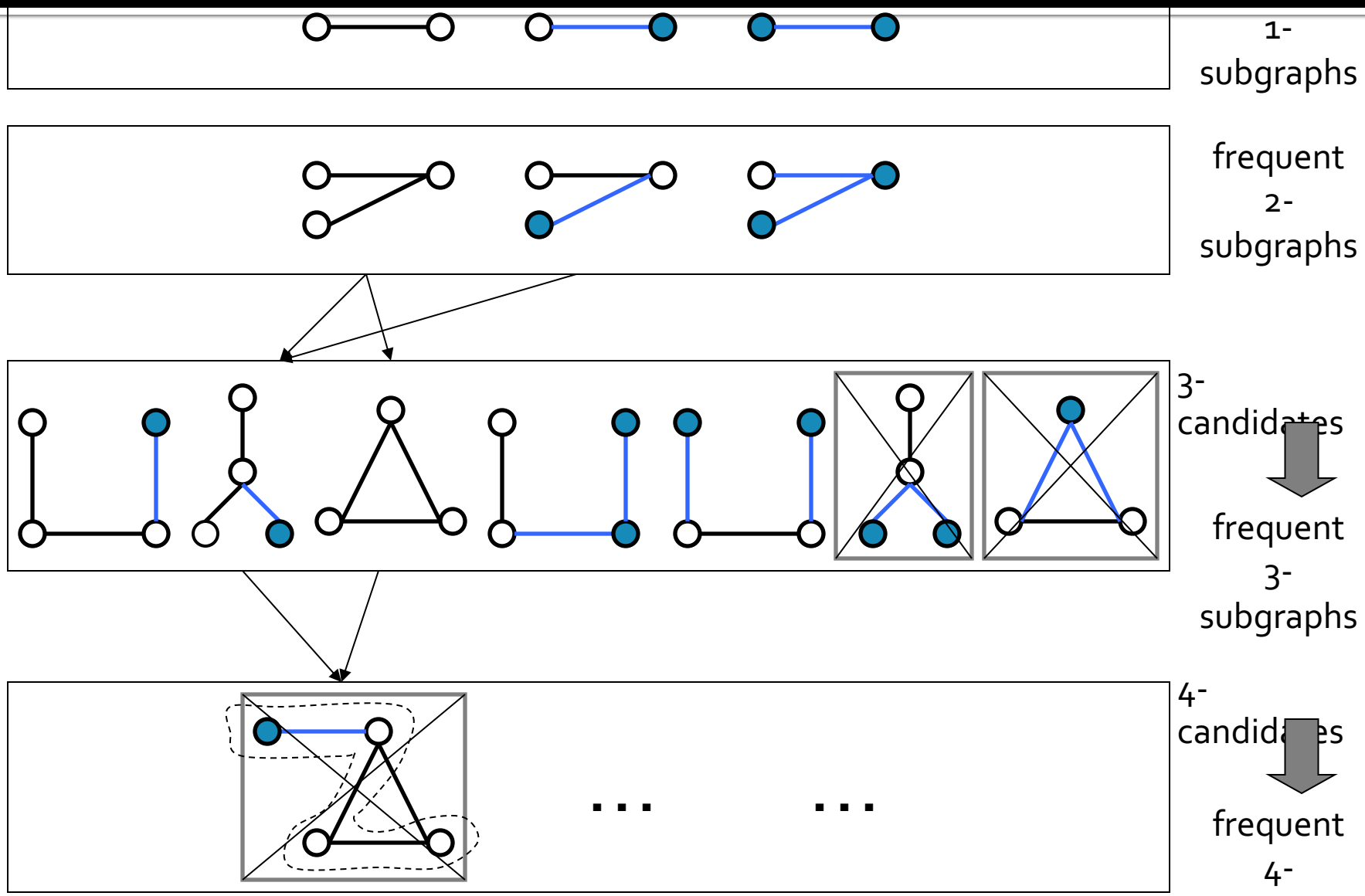
3-candidates:



4-candidates:



- Every (k-1)-subgraph must be frequent.
- For all the (k-1)-subgraphs of a given k-candidate, check if downward closure property holds


1-
subgraphs
frequent
2-
subgraphs
3-
candidates
frequent
3-
subgraphs
4-
candidates
frequent
4-
subgraphs

Computational Challenges

Simple operations become complicated & expensive when dealing with graphs…

- Candidate generation
  - To determine if we can join two candidates, we need to perform subgraph isomorphism to determine if they have a common subgraph.
  - There is no obvious way to reduce the number of times that we generate the same subgraph.
  - Need to perform graph isomorphism for redundancy checks.
  - The joining of two frequent subgraphs can lead to multiple candidate subgraphs.
- Candidate pruning
  - To check downward closure property, we need subgraph isomorphism.
- Frequency counting
  - Subgraph isomorphism for checking containment of a frequent subgraph

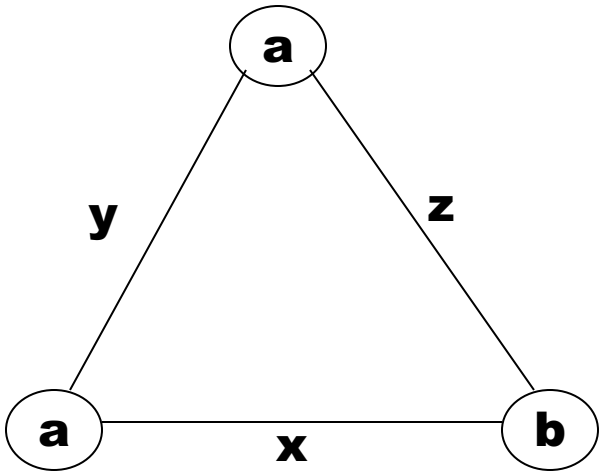
Computational Challenges

- Key to FSG's computational efficiency:
  - Uses an efficient algorithm to determine a *canonical labeling* of a graph and use these *"strings"* to perform identity checks (simple comparison of strings!).
  - Uses a sophisticated candidate generation algorithm that reduces the number of times each candidate is generated.
  - Uses an augmented TID-list based approach to speedup frequency counting.

FSG: Canonical representation for graphs (based on adjacency matrix)

Code(M_1) = "abyzx"
Code(M_2) = "abaxyz"

Graph G:



M_1 :

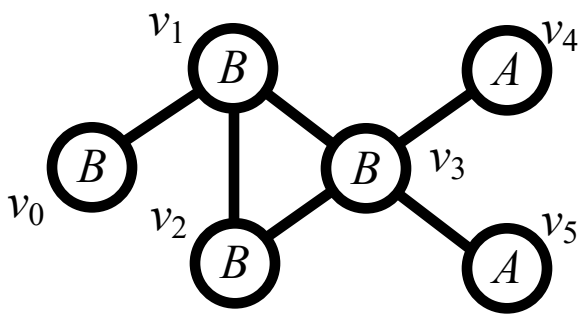
	a	a	b
a		y	z
a	y		x
b	z	x	

M_2 :

	a	b	a
a		x	y
b	x		z
a	y	z	

Code(G) = min{ code(M) | M is adj. Matrix}

Canonical labeling



		v_3	v_1	v_2	v_4	v_5	v_0
		B	B	B	A	A	B
v_3	B		1	1	1	1	
v_1	B	1		1			1
v_2	B	1	1				
v_4	A	1					
v_5	A	1					
v_0	B		1				

Label = "1 11 100 1000 01000"

		v_0	v_1	v_2	v_3	v_4	v_5
		B	B	B	B	A	A
v_0	B		1				
v_1	B	1		1	1		
v_2	B		1		1		
v_3	B		1	1		1	1
v_4	A				1		
v_5	A				1		

Label = "1 01 011 0001 00010"

FSG: Finding the canonical labeling

- The problem is as complex as graph isomorphism, but FSG suggests some heuristics to speed it up such as:
  - Vertex Invariants (e.g. degree)
  - Neighbor lists
  - Iterative partitioning

Another FSG Heuristic: frequency counting

Transactions

$g^{k-1}_1, g^{k-1}_2 \subset T1$

$g^{k-1}_1 \subset T2$

$g^{k-1}_1, g^{k-1}_2 \subset T3$

$g^{k-1}_2 \subset T6$

$g^{k-1}_1 \subset T8$

$g^{k-1}_1, g^{k-1}_2 \subset T9$

Frequent subgraphs

$TID(g^{k-1}_1) = \{ 1, 2, 3, 8, 9 \}$

$TID(g^{k-1}_2) = \{ 1, 3, 6, 9 \}$

Candidate

$c^k = join(g^{k-1}_1, g^{k-1}_2)$

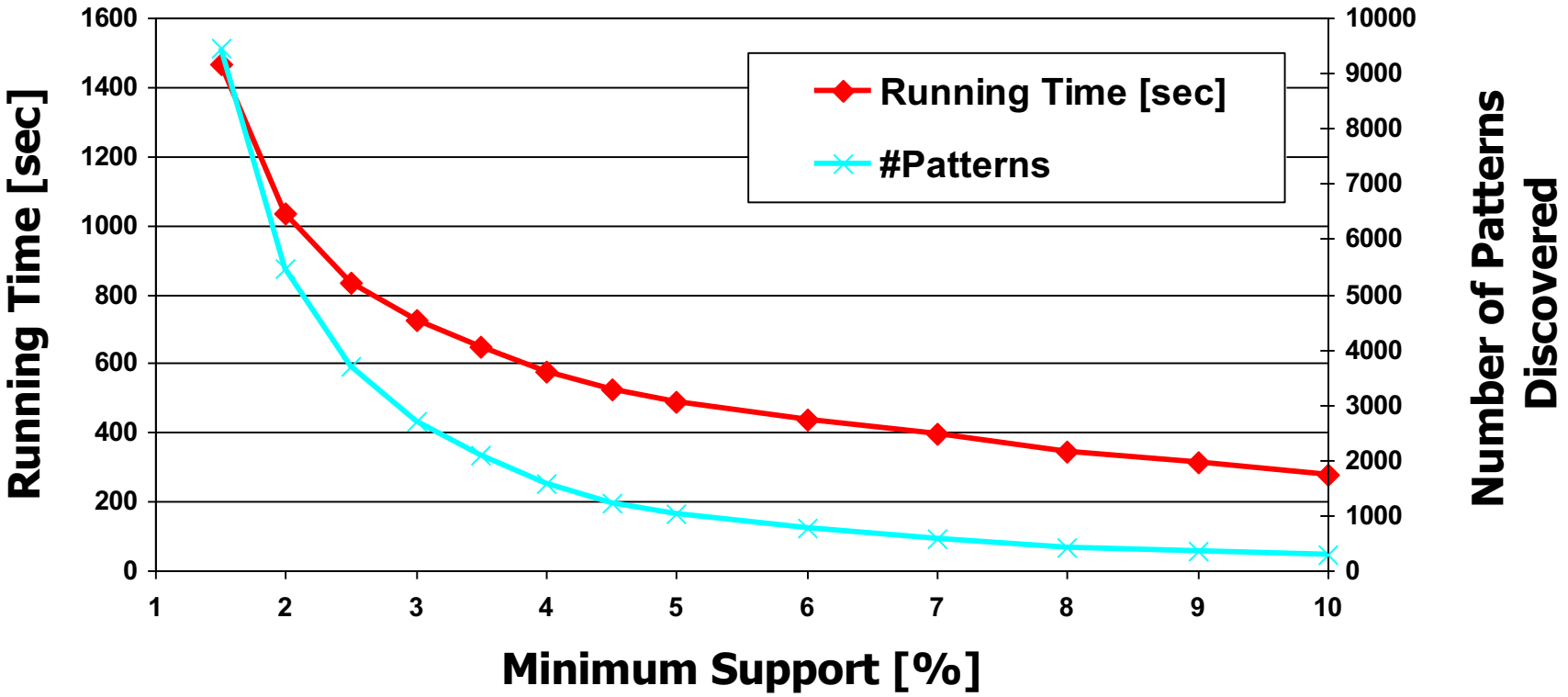
$TID(c^k) \subset TID(g^{k-1}_1) \cap TID(g^{k-1}_2)$

$\Downarrow$

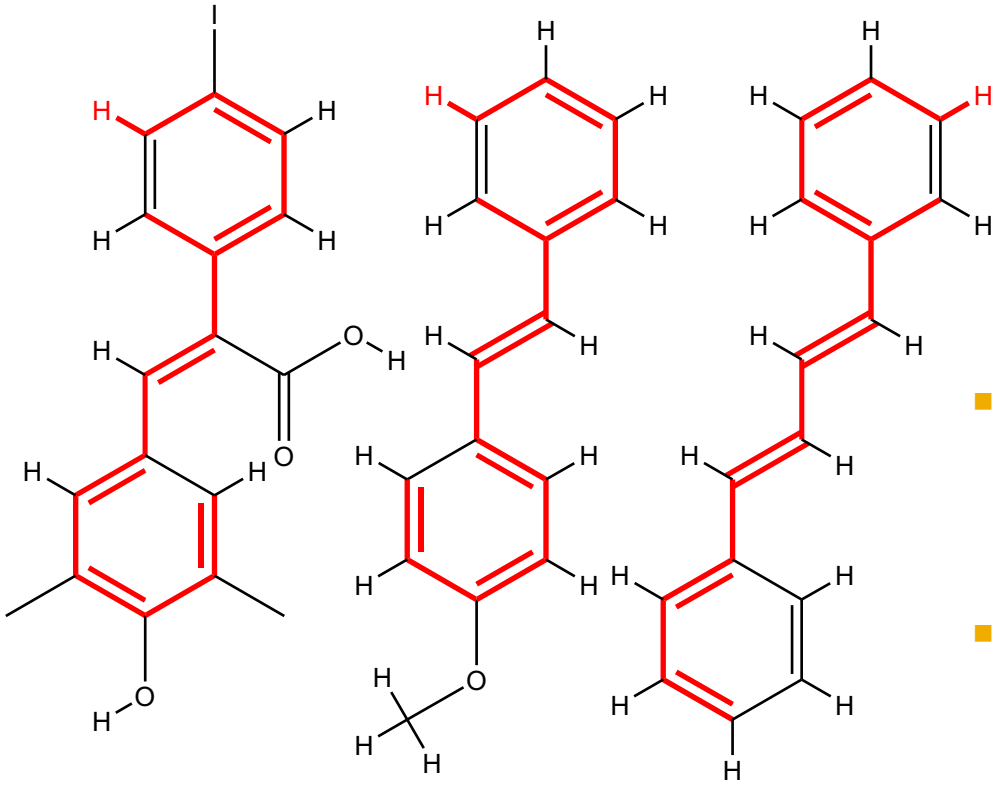
$TID(c^k) \subset \{ 1, 3, 9\}$

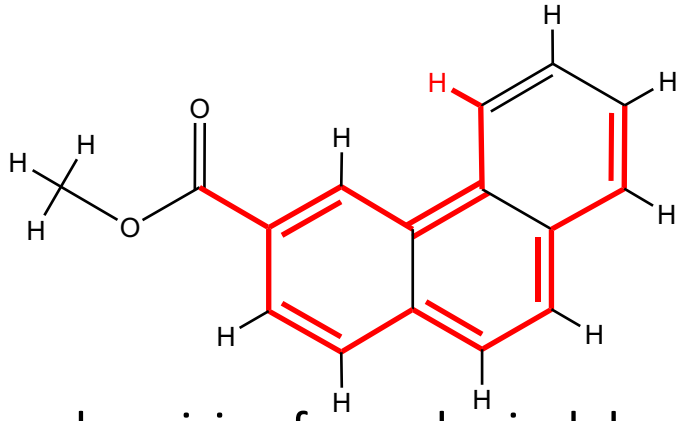
- Perform subgraph-iso to T1, T3 and T9 with c^k and determine $TID(c^k)$
- Note, TID lists require a lot of memory.

FSG performance on DTP Dataset (chemical compounds)



Topology Is Not Enough (Sometimes)

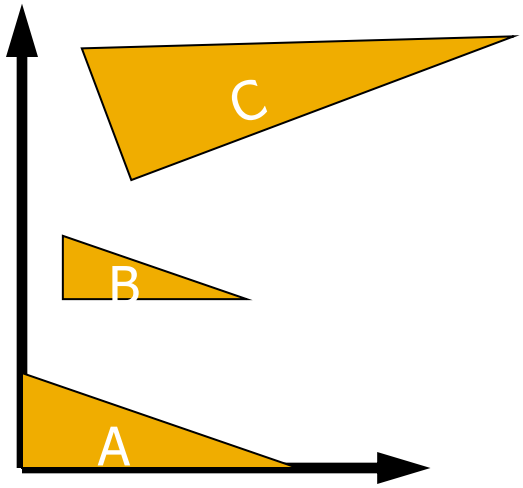


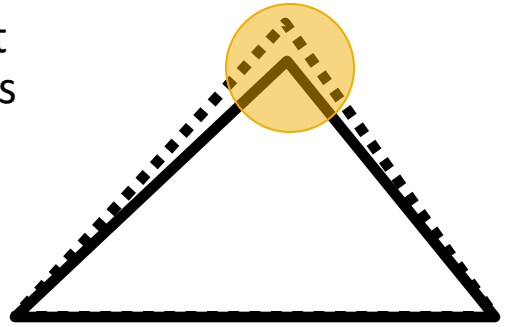


- Graphs arising from physical domains have a strong geometric nature.
  - This geometry must be taken into account by the data-mining algorithms.
- Geometric graphs.
  - Vertices have physical 2D and 3D coordinates associated with them.

gFSG—Geometric Extension Of FSG (Kuramochi & Karypis ICDM 2002)

- Same input and same output as FSG
  - Finds frequent geometric connected subgraphs
- Geometric version of (sub)graph isomorphism
  - The mapping of vertices can be translation, rotation, and/or scaling invariant.
  - The matching of coordinates can be inexact as long as they are within a tolerance radius of *r.*
    - *R*-tolerant geometric isomorphism.





Different Approaches for GM

- Apriori Approach
  - FSG
  - Path Based
- DFS Approach
  - gSpan
- Greedy Approach
  - Subdue

Y. Xifeng and H. Jiawei

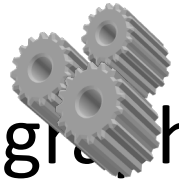
gSpan: Graph-Based Substructure Pattern Mining

ICDM, 2002.

part1:

Define the Tree Search Space (TSS)

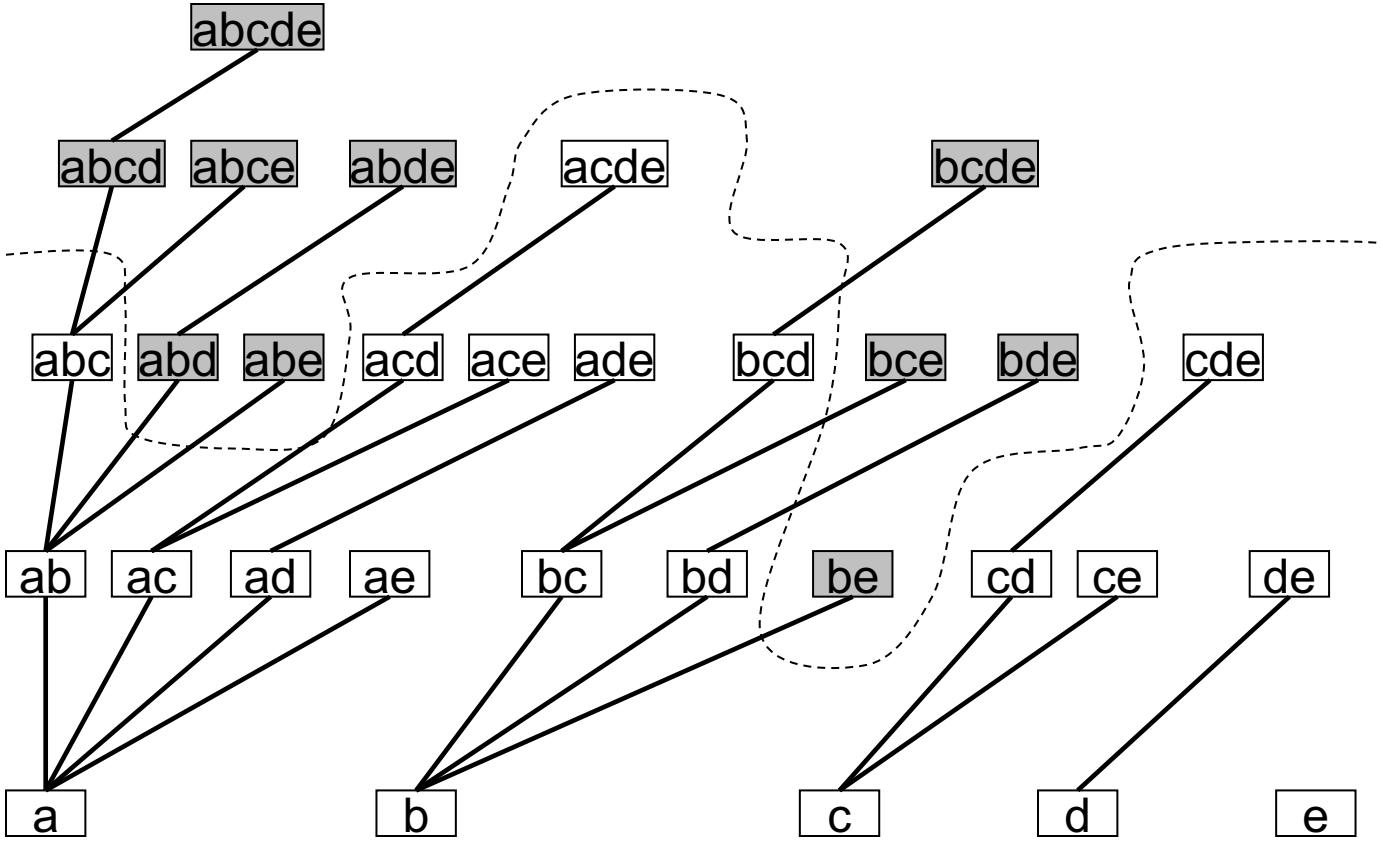
Part2:



Find all frequent graphs
by exploring TSS

Motivation: DFS exploration wrt. itemsets.

Itemset search space – prefix based



Motivation for TSS

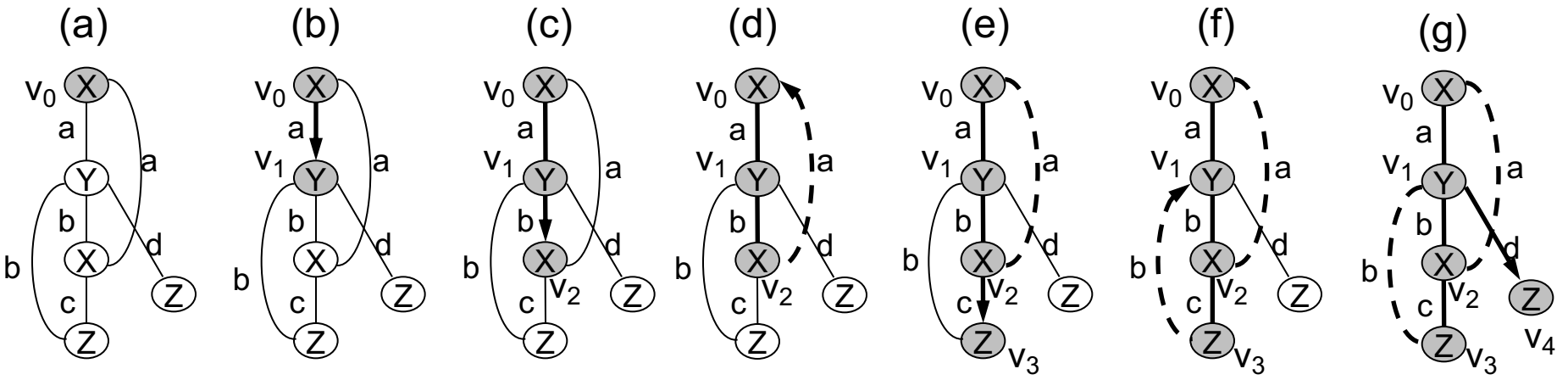
- **Canonical representation** of itemset is obtained by a complete order over the items.
- Each possible itemset appear in TSS exactly once - no duplications or omissions.
- **Properties of Tree search space**
  - for each k-label, its parent is the k-1 prefix of the given k-label
  - The relation among siblings is in ascending lexicographic order.

DFS Code representation

- Map each graph (2-Dim) to a sequential DFS Code (1-Dim).
- Lexicographically order the codes.
- Construct TSS based on the lexicographic order.

DFS Code construction

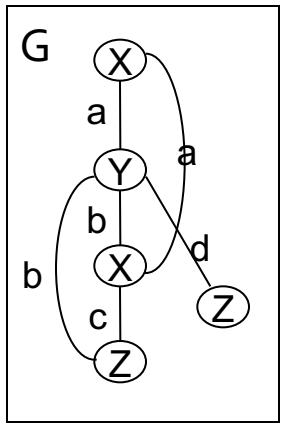
- Given a graph G. for each Depth First Search over graph G, construct the corresponding DFS-Code.

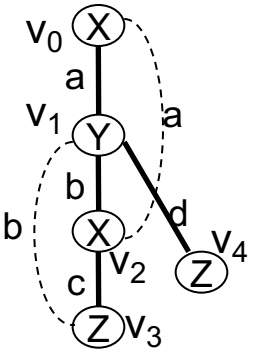


(0,1,X,a,Y)	(1,2,Y,b,X)	(2,0,X,a,X)	(2,3,X,c,Z)	(3,1,Z,b,Y)	(1,4,Y,d,Z)

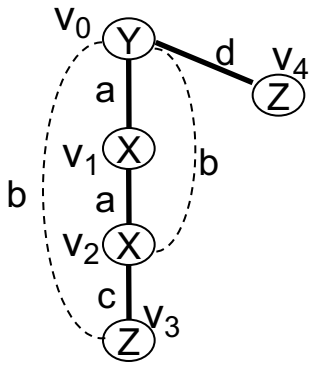
Single graph, several DFS-Codes

	(a)	(b)	(c)
1	(0, 1, X, a, Y)	(0, 1, Y, a, X)	(0, 1, X, a, X)
2	(1, 2, Y, b, X)	(1, 2, X, a, X)	(1, 2, X, a, Y)
3	(2, 0, X, a, X)	(2, 0, X, b, Y)	(2, 0, Y, b, X)
4	(2, 3, X, c, Z)	(2, 3, X, c, Z)	(2, 3, Y, b, Z)
5	(3, 1, Z, b, Y)	(3, 0, Z, b, Y)	(3, 0, Z, c, X)
6	(1, 4, Y, d, Z)	(0, 4, Y, d, Z)	(2, 4, Y, d, Z)

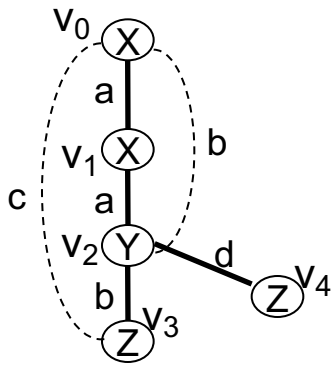




(a)



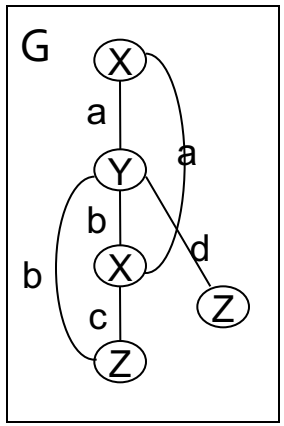
(b)

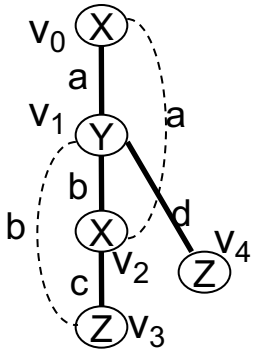


(c)

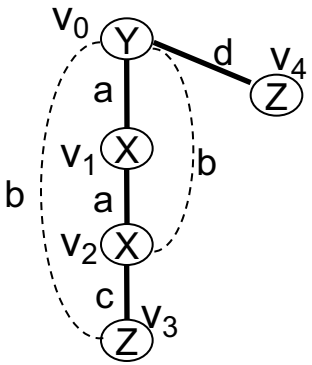
Single graph - single Min DFS-code!

	(a)	(b)	(c) Min DFS-Code
1	(0, 1, X, a, Y)	(0, 1, Y, a, X)	**(0, 1, X, a, X)**
2	(1, 2, Y, b, X)	(1, 2, X, a, X)	**(1, 2, X, a, Y)**
3	(2, 0, X, a, X)	(2, 0, X, b, Y)	**(2, 0, Y, b, X)**
4	(2, 3, X, c, Z)	(2, 3, X, c, Z)	**(2, 3, Y, b, Z)**
5	(3, 1, Z, b, Y)	(3, 0, Z, b, Y)	**(3, 0, Z, c, X)**
6	(1, 4, Y, d, Z)	(0, 4, Y, d, Z)	**(2, 4, Y, d, Z)**

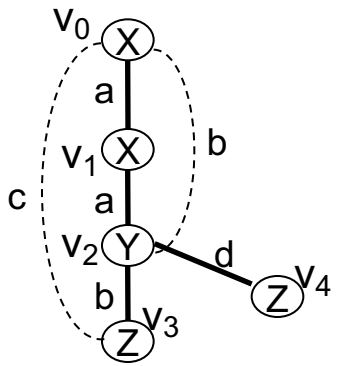




(a)



(b)



(c)

Minimum DFS-Code

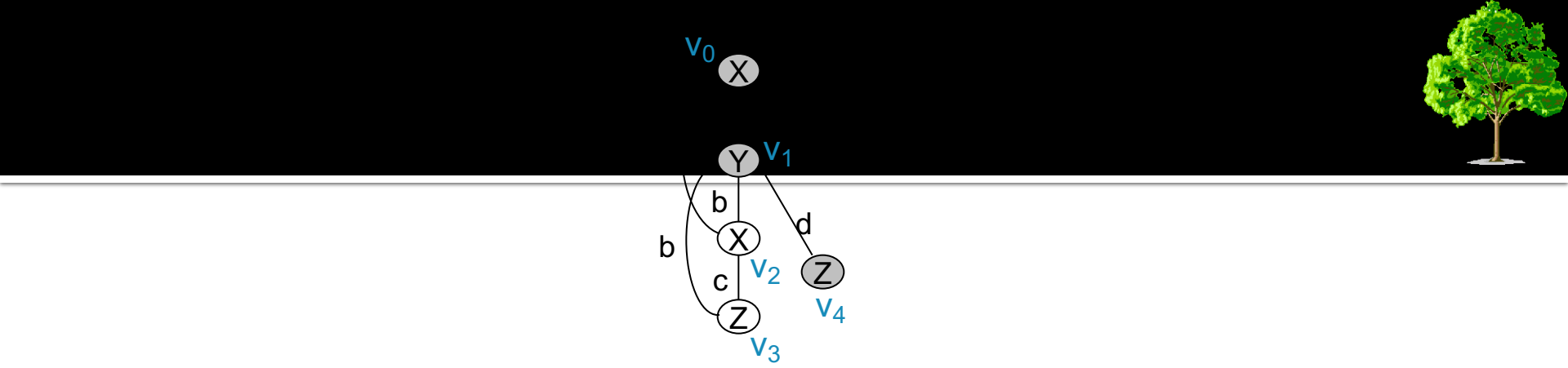
- The minimum DFS code *min(G)*, in DFS lexicographic order, is a canonical representation of graph *G*.

- Graphs *A* and *B* are isomorphic if and only if:

$$min(A) = min(B)$$

DFS-Code Tree: parent-child relation

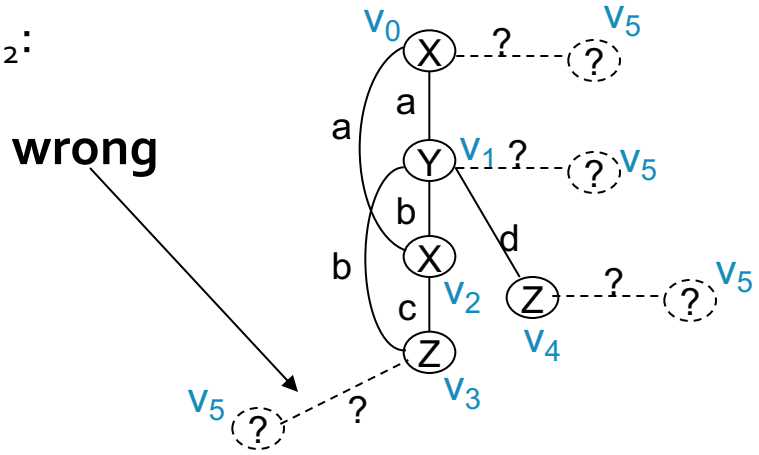
- If $\min(G_1) = \{ a_0, a_1, \ldots, a_n\}$
  and $\min(G_2) = \{ a_0, a_1, \ldots, a_n, b\}$
  - G_1 is parent of G_2
  - G_2 is child of G_1
- A valid DFS code requires that **b** grows from a vertex on the rightmost path (inherited property from the DFS search).

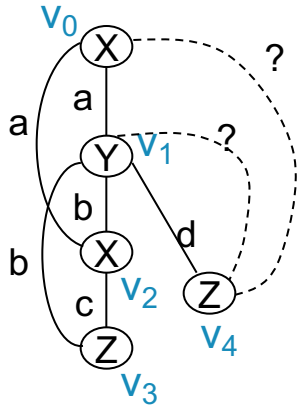


Min(g) = (0,1,X,a,Y) (1,2,Y,b,X) (2,0,X,a,X) (2,3,X,c,Z) (3,1,Z,b,Y) (1,4,Y,d,Z)

A child of graph G_1 *must grow edge from rightmost path of* G_1 (necessary condition)

Graph G_2:



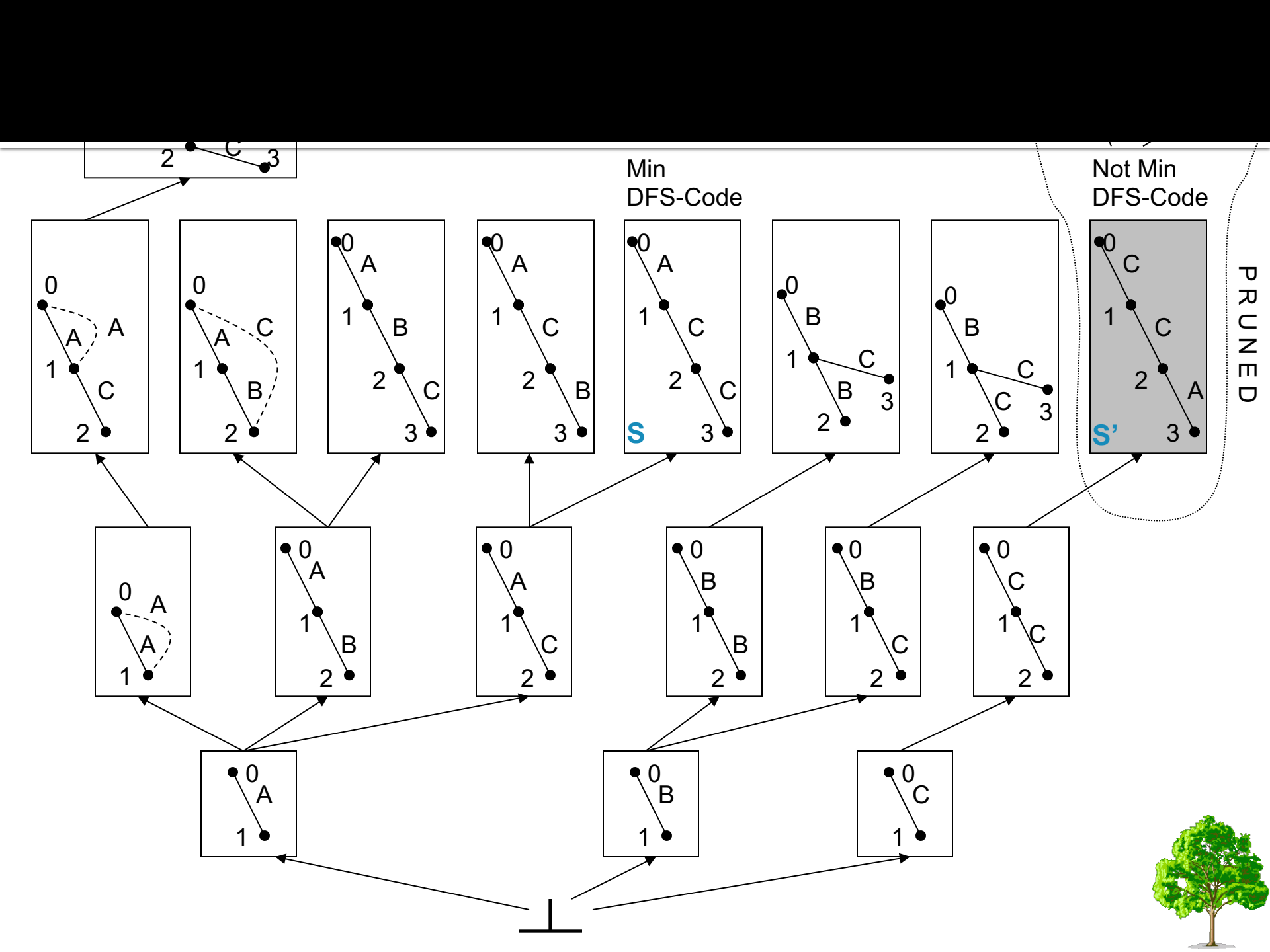


Forward EDGE

Backward EDGE

Search space: DFS code Tree

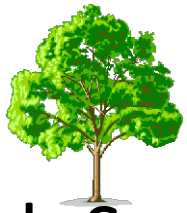
- Organize DFS Code nodes as parent-child.
- Sibling nodes organized in ascending DFS lexicographic order.
- *InOrder* traversal follows DFS lexicographic order!


Min
DFS-Code
Not Min
DFS-Code
PRUNED
S
S'

Tree pruning

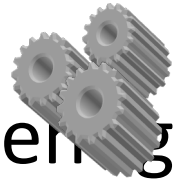
- All of the descendants of infrequent node are infrequent also.
- All of the descendants of a not minimal DFS code are also not minimal DFS codes.

part1:



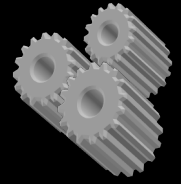
defining the Tree Search Space (TSS)

Part2:



gSpan Finds all frequent graphs
by Exploring TSS

gSpan Algorithm



gSpan(*D*, *F*, *g*)

1: if $g \neq \min(g)$

    return;

2: $F \leftarrow F \cup \{ g \}$

3: children(g) $\leftarrow$ [generate all g' potential children with one edge growth]*

4: Enumerate(D, g, children(g))

5: for each $c \in$ children(g)

    if support(c) $\geq$ #minSup

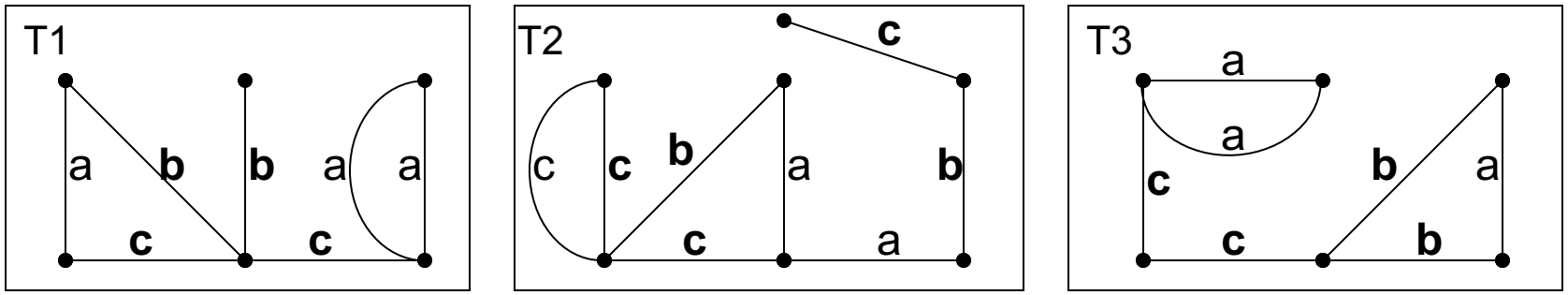
        SubgraphMining (D, F, c)

---

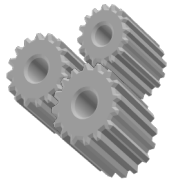
* gSpan improve this line

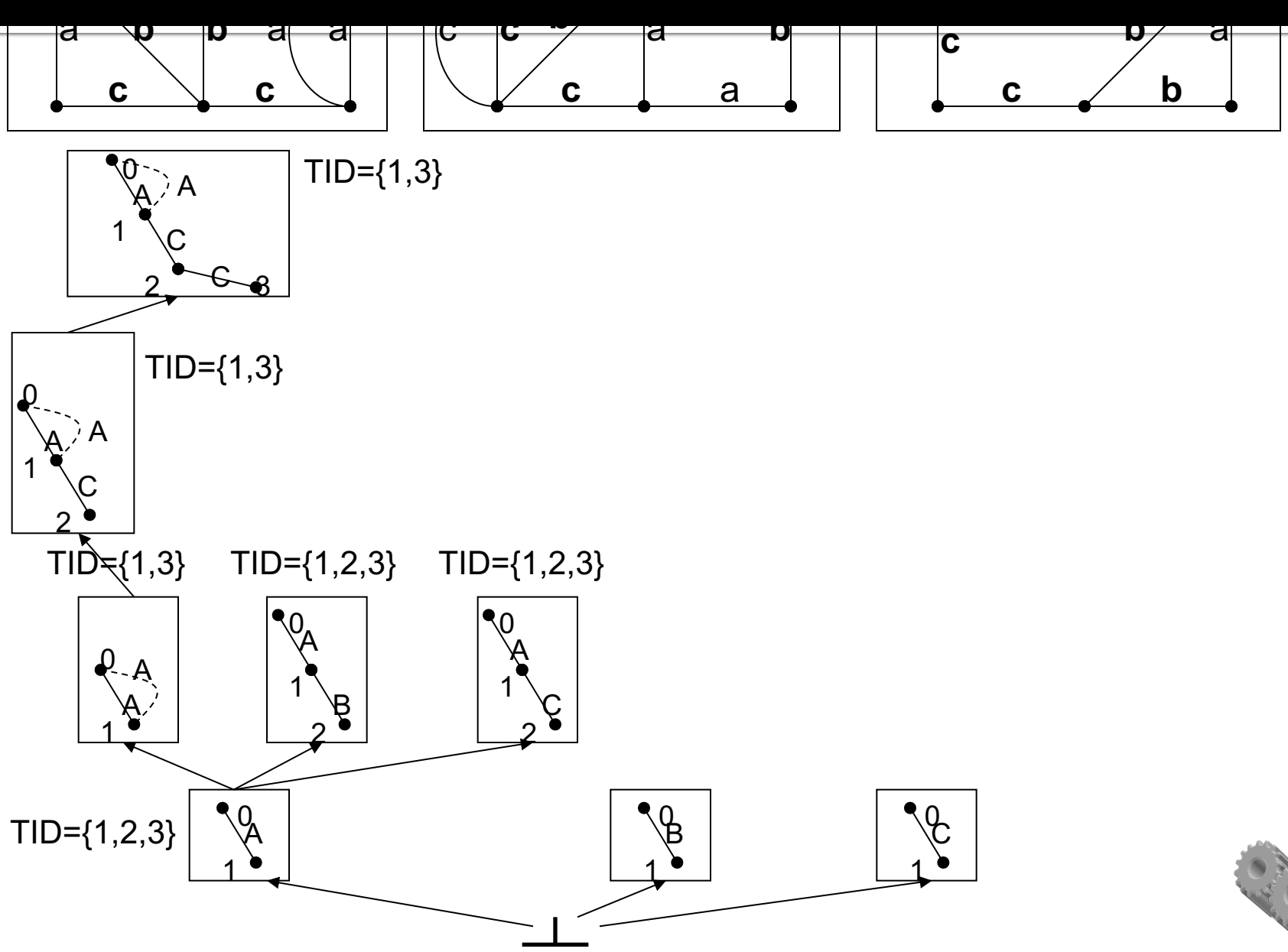
Example

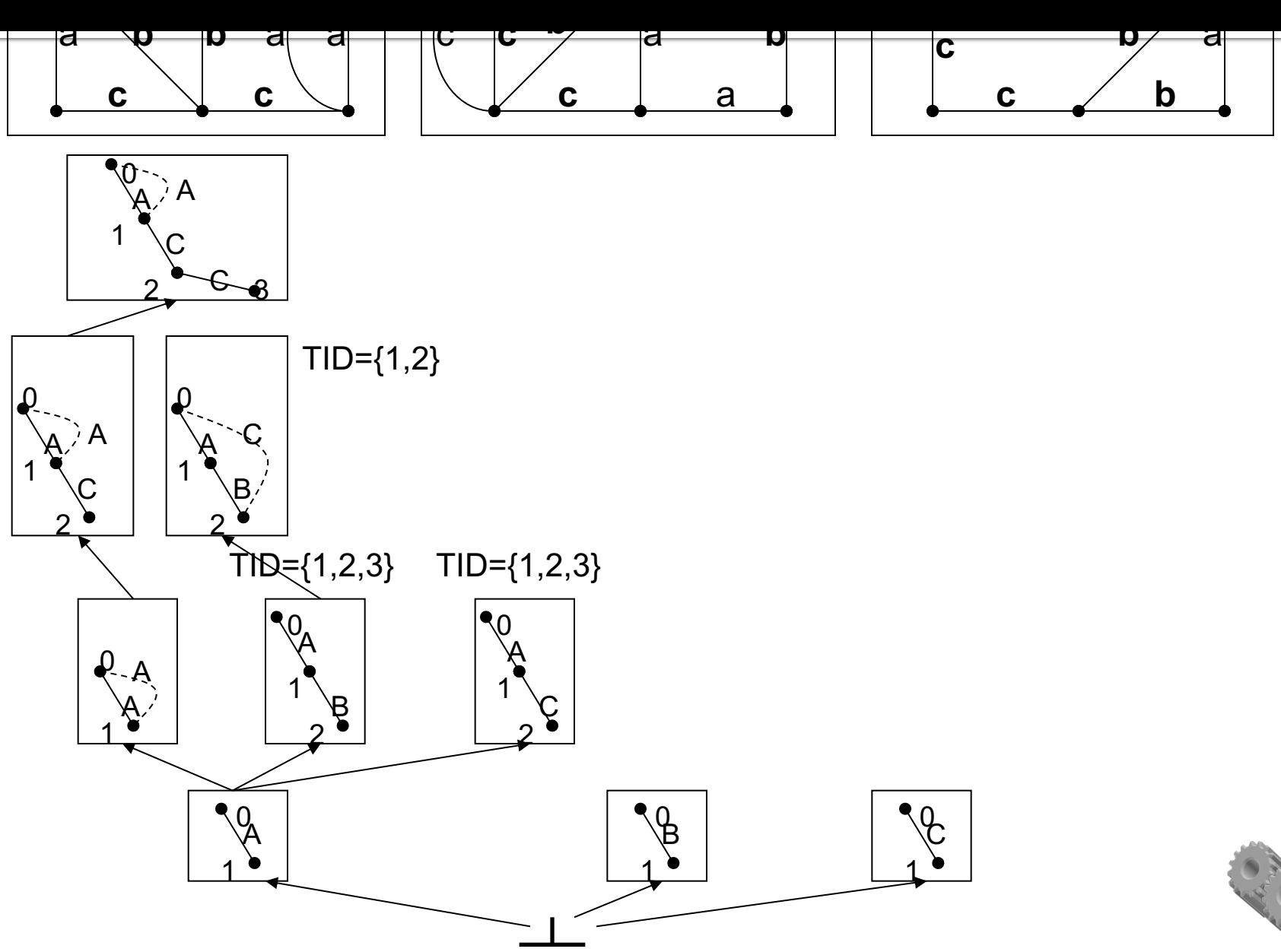
Given: database *D*

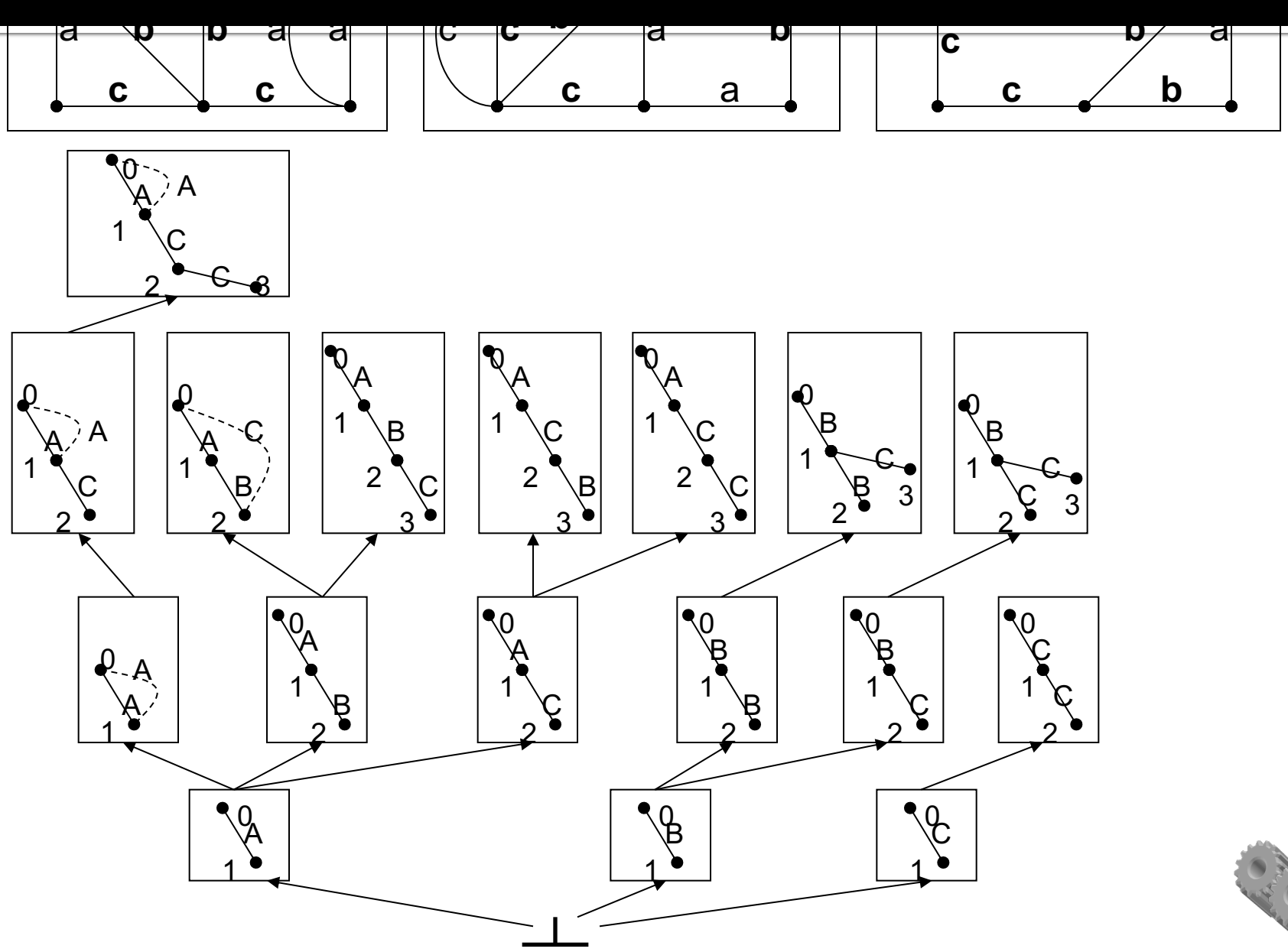


Task: Mine all frequent subgraphs with support ≥ 2 (#minSup)




TID={1,3}
TID={1,3}
TID={1,3}
TID={1,2,3}
TID={1,2,3}
TID={1,2,3}


a b b a a
c c
c c a b
c a
c b a
c b
0 A A
1 C
2 C 3
TID={1,2}
0 A A
1 C
2
0 A C
1 B
2
TID={1,2,3}
TID={1,2,3}
0 A
A
1
0 A
1 B
2
0 A
1 C
2
0 A
1
0 B
1
0 C
1


a b b a a
c c
c c a b
c a
c b a
c b
0 A A 1 C 2 C 3
0 A A 1 C 2
0 A C 1 B 2
0 A 1 B 2 C 3
0 A 1 C 2 B 3
0 A 1 C 2 C 3
0 B 1 C B 2 3
0 B 1 C C 2 3
0 A A 1
0 A 1 B 2
0 A 1 C 2
0 B 1 B 2
0 B 1 C 2
0 C 1 C 2
0 A 1
0 B 1
0 C 1

gSpan Performance

- On synthetic datasets it was 6-10 times faster than FSG.
- On Chemical compounds datasets it was 15-100 times faster!
- But this was comparing to OLD versions of FSG!

Different Approaches for GM

- Apriori Approach
  - FSG
  - Path Based
- DFS Approach
  - gSpan
- Greedy Approach
  - Subdue

D. J. Cook and L. B. Holder

Graph-Based Data Mining

Tech. report, Department of CS Engineering, 1998

Graph Pattern Explosion Problem

- If a graph is frequent, all of its subgraphs are frequent – **the Apriori property**
- An n-edge frequent graph may have 2^n subgraphs.
- Among 422 chemical compounds which are confirmed to be active in an AIDS antiviral screen dataset, there are 1,000,000 frequent graph patterns if the minimum support is 5%.

Subdue algorithm

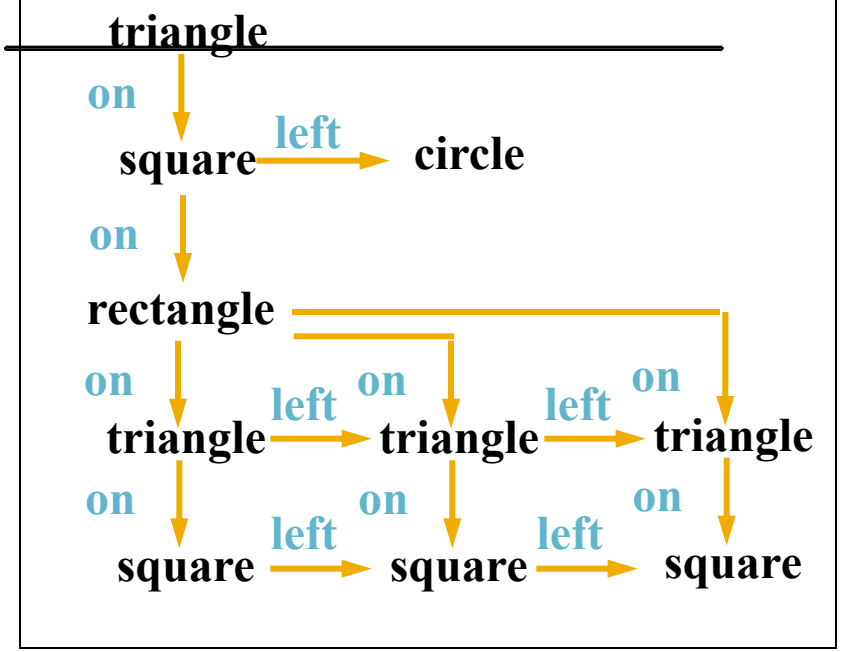
- A greedy algorithm for finding some of the most prevalent subgraphs.
- This method is not complete, i.e. it may not obtain all frequent subgraphs, although it pays in fast execution.

Subdue algorithm (Cont.)

- It discovers substructures that compress the original data and represent structural concepts in the data.
- Based on *Beam Search* - like BFS it progresses level by level. Unlike BFS, however, beam search moves downward only through the best *W* nodes at each level. The other nodes are ignored.

Step 1: Create substructure for each unique vertex label

DB:

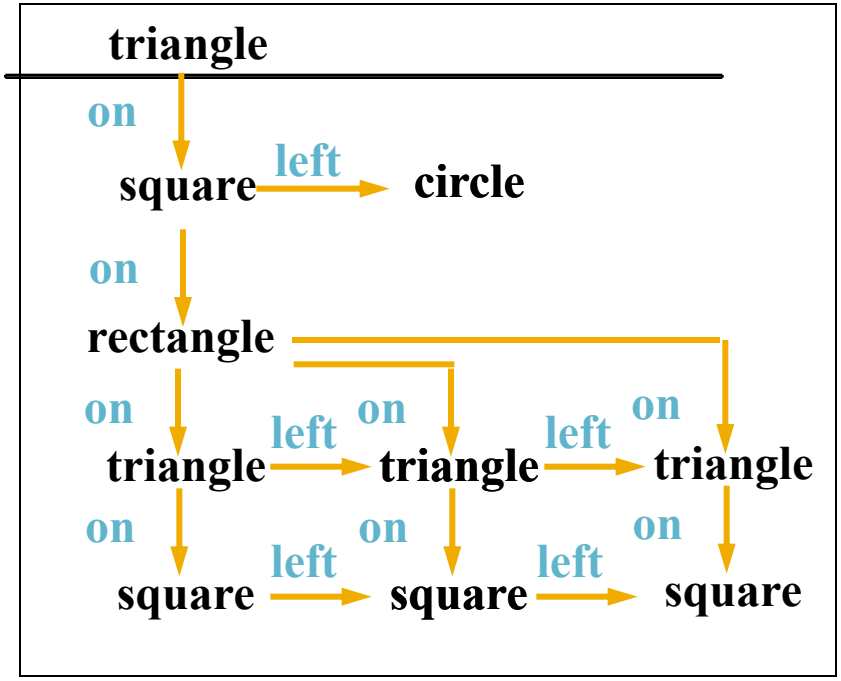


Substructures:

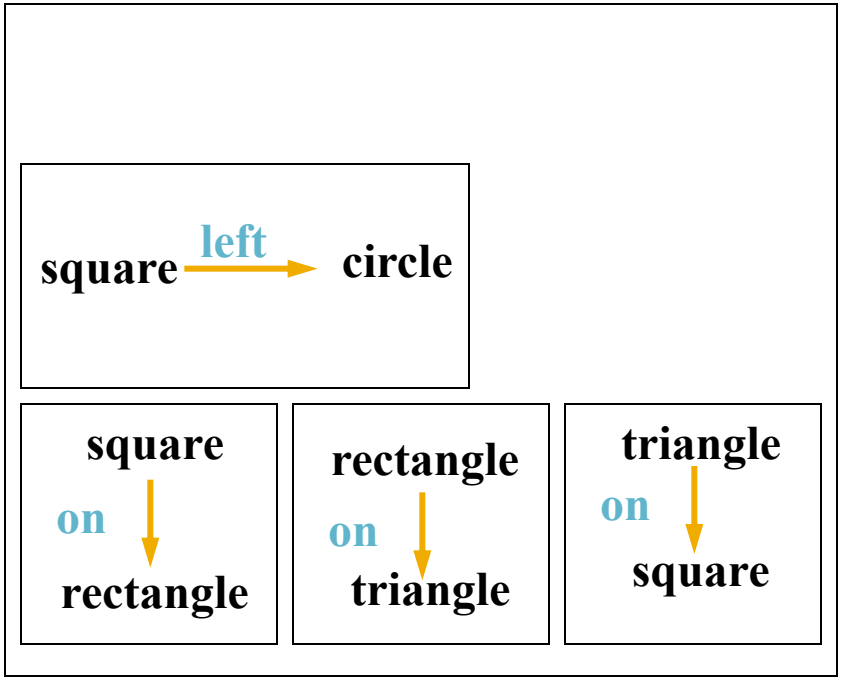
triangle (4)
square (4)
circle (1)
rectangle (1)

Step 2: Expand best substructure by an edge or edge and neighboring vertex

DB:



Substructures:



Step 3: Keep only best substructures on queue (specified by beam width).

Step 4: Terminate when queue is empty or when the number of discovered substructures is greater than or equal to the limit specified.

Step 5: Compress graph and repeat to generate hierarchical description.

Outline

- Introduction
- Motivation and applications for Graph mining
- Mining Frequent Subgraphs – Transaction setting
  - BFS/Apriori Approach (FSG and others)
  - DFS Approach (gSpan and others)
  - Greedy Approach
- Mining Frequent Subgraphs – Single graph setting
  - The support issue
  - Path mining algorithm
  - Constraint-based mining

Most existing algorithms use a transaction setting approach.
That is, if a pattern appears in a transaction even multiple times it is counted as 1 (FSG, gSPAN).

What if the entire database is a single graph?
This is called **single graph setting**.

We need a different support definition!

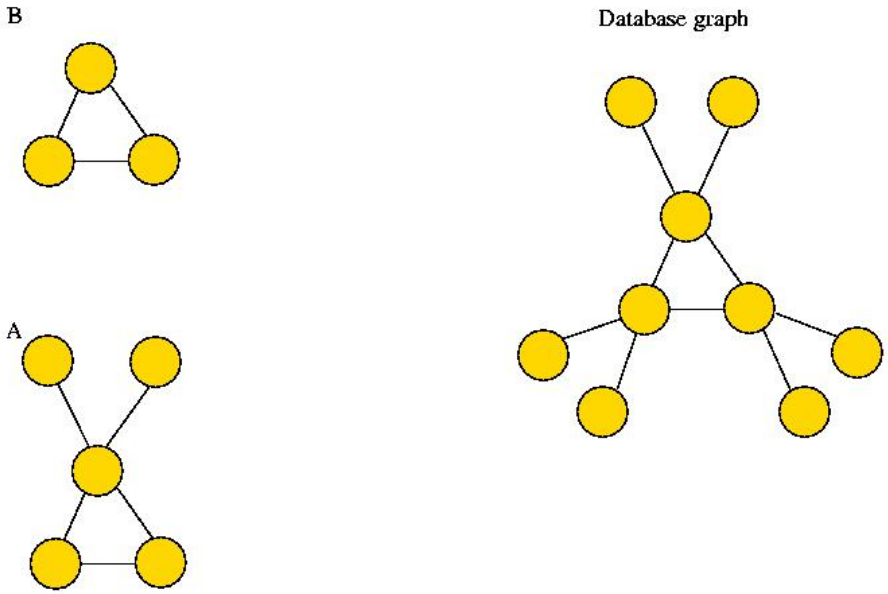
Single graph setting - Motivation

- Often the input is a single large graph.
- **Examples:**
  - The web or portions of it.
  - A social network (e.g. a network of users communicating by email at BGU).
  - A large XML database such as DBLP or Movies database.
- Mining large graph databases is very useful.

Support issue

Support measure is **admissible** if for any pattern P and any sub-pattern $Q \subset P$ support of P is not larger than support of Q.

Problem: the number of pattern appearances is not good!



An instance graph of pattern P in database graph D is a graph whose nodes are pattern instances in D and they are connected by an edge when corresponding instances share an edge.

Operations on instance graph:

- **clique contraction:** replace clique C by a single node c. Only the nodes adjacent to each node of C may be adjacent to c.

- **node expansion:** replace node v by a new subgraph whose nodes may or may not be adjacent to the nodes adjacent to v.

- **node addition:** add a new node to the graph and arbitrary edges between the new node and the old ones.

- **edge removal :** remove an edge.

The main result

Theorem. A support measure S is an admissible support measure if and only if it is non-decreasing on instance graph of every pattern P under clique contraction, node expansion, node addition and edge removal.

$$\text{MIS} = \frac{\text{Maximum independent set size of instance graph}}{\text{Number of edges in the database graph}}$$

- Goal: find all frequent connected subgraphs
  of a database graph.

- Basic approach: Apriori or BFS.
  - The basic building block is a path not an edge.
  - This works since any graph can be decomposed into edge-disjoint paths.

- Result: faster convergence of the algorithm.

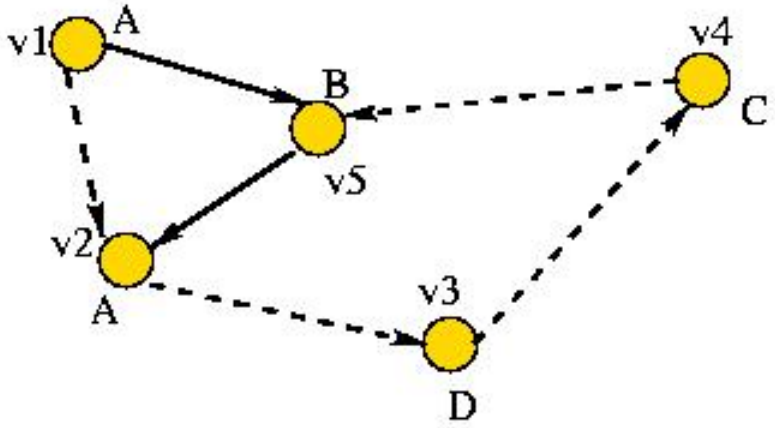
Path-based mining algorithm

- The algorithm uses paths as basic building blocks for pattern construction.
- It starts with one-path graphs and combines them into 2-, 3- etc. path graphs.
- The combination technique does not use graph operations and is easy to implement.
- Path number of a graph is computed in linear time: it is the number of odd-degree vertices divided by two.
- Given minimal path cover P, removal of one path creates a graph with minimal path cover size $|P|$-1.
- There exist at least two paths in P whose removal leaves the graph connected.

More than one path cover for graph

1. Define a **descriptor** of each path based on node labels and node degrees.
2. Use **lexicographical order** among descriptors to compare between paths.
3. One graph can have several minimal path covers.
4. We only use path covers that are **minimal** w.r.t. lexicographical order.
5. Removal of path from a lexicographically minimal path cover leaves the cover lexicographically minimal.

P_1 = v1,v2,v3,v4,v5 P_2 = v1,v5,v2



Desc (P_1) = <A,0,1>,<A,1,1>,<B,1,0>,<C,1,1>,<D,1,1>
Desc (P_2) = <A,0,1>,<A,1,0>,<B,1,1>

Path mining algorithm

- Phase 1: find all frequent 1-path graphs.
- Phase 2: find all frequent 2-path graphs by "joining" frequent 1-path graphs.
- Phase 3: find all frequent k-path graphs, $k \geq 3$, by "joining" pairs of frequent (k-1)-path graphs.

Main challenge: "join" must ensure soundness and completeness of the algorithm.

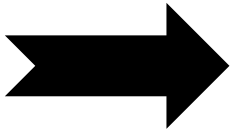
Graph composed from 3 paths:

Node	P_1	P_2	P_3
v1	a1	⊥	⊥
v2	a2	b2	⊥
v3	a3	⊥	⊥
v4	⊥	b1	⊥
v5	⊥	b3	c3
v6	⊥	⊥	c1
v7	⊥	⊥	c2

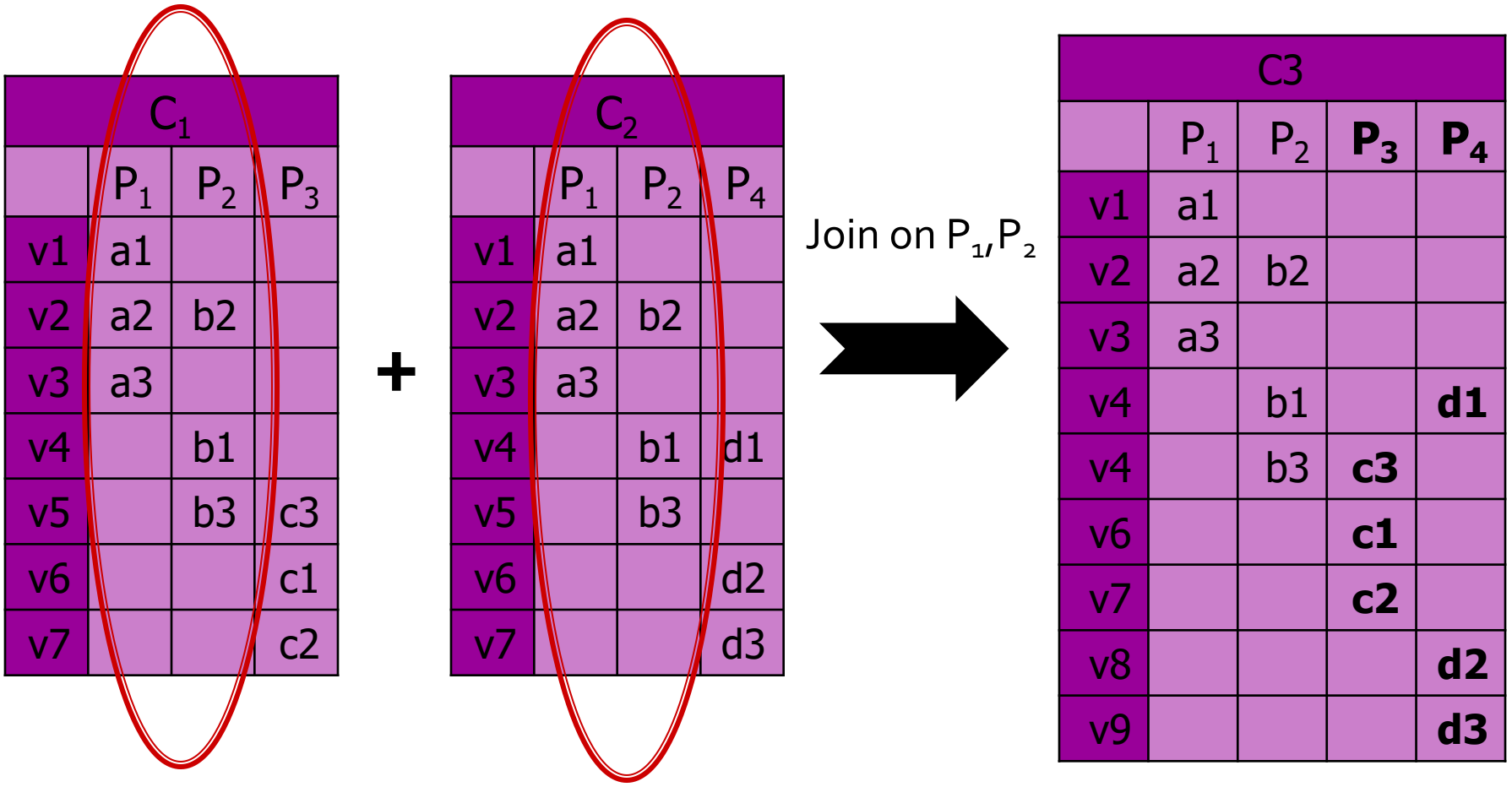
Removing path from table

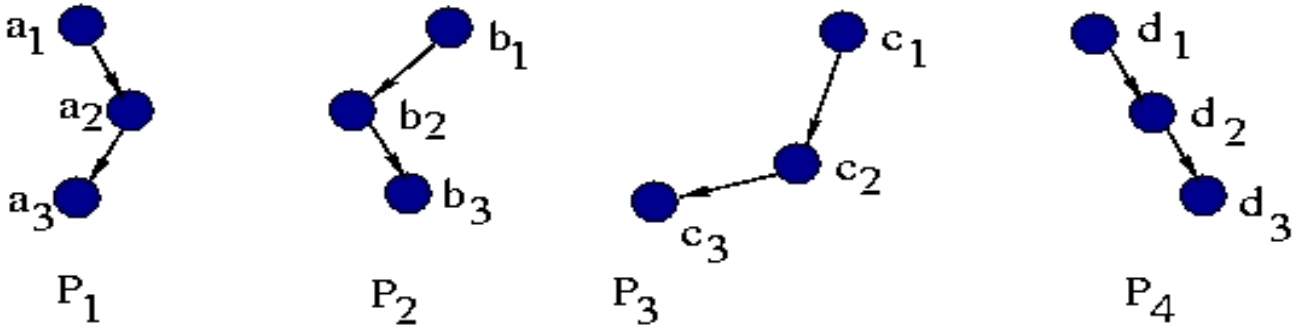
Node	P_1	P_2	P_3
v1	a1	⊥	⊥
v2	a2	b2	⊥
v3	a3	⊥	⊥
v4	⊥	b1	⊥
v5	⊥	b3	c3
v6	⊥	⊥	c1
v7	⊥	⊥	c2

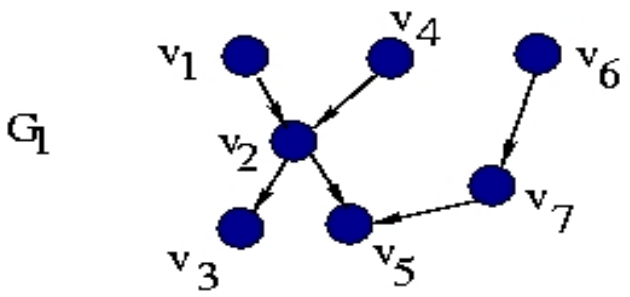
delete P_3

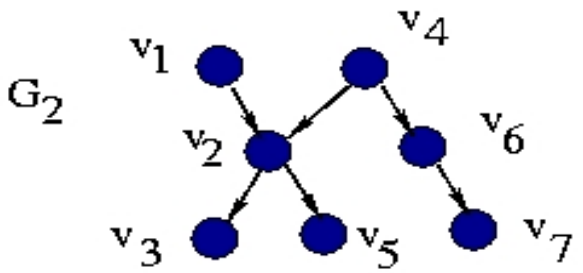


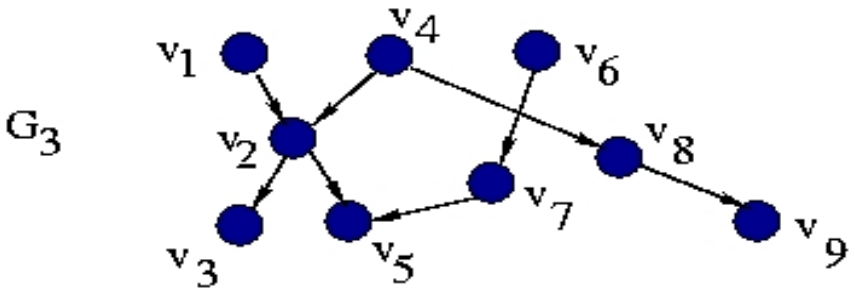
Node	P_1	P_2
v1	a1	⊥
v2	a2	b2
v3	a3	⊥
v4	⊥	b1
v5	⊥	b3


C1
P1 P2 P3
v1 a1
v2 a2 b2
v3 a3
v4 b1
v5 b3 c3
v6 c1
v7 c2
+
C2
P1 P2 P4
v1 a1
v2 a2 b2
v3 a3
v4 b1 d1
v5 b3
v6 d2
v7 d3
Join on P1, P2
C3
P1 P2 P3 P4
v1 a1
v2 a2 b2
v3 a3
v4 b1 d1
v4 b3 c3
v6 c1
v7 c2
v8 d2
v9 d3


a1
a2
a3
b1
b2
b3
c1
c2
c3
d1
d2
d3
P1
P2
P3
P4


G1
v1
v2
v3
v4
v5
v6
v7


G2
v1
v2
v3
v4
v5
v6
v7


G3
v1
v2
v3
v4
v5
v6
v7
v8
v9

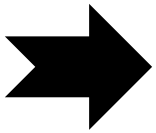
- We need to construct a frequent n-path graph G on paths P1,…,Pn.
- We have two frequent (n-1)-path graphs, G1 on paths P1,…,Pn-1 and G2 on paths P2,…,Pn.
- The **sum** of G1 and G2 will give us n-path graph G’ on paths P1,…,Pn.
- G’=G if P1 and Pn have no common node that belongs solely to them.
- A frequent 2-path graph H containing P1 and Pn exactly as they appear in G exists if G is frequent.
- Let us join the nodes of P1 and Pn in G’ according to H.

This is the **splice** operation!

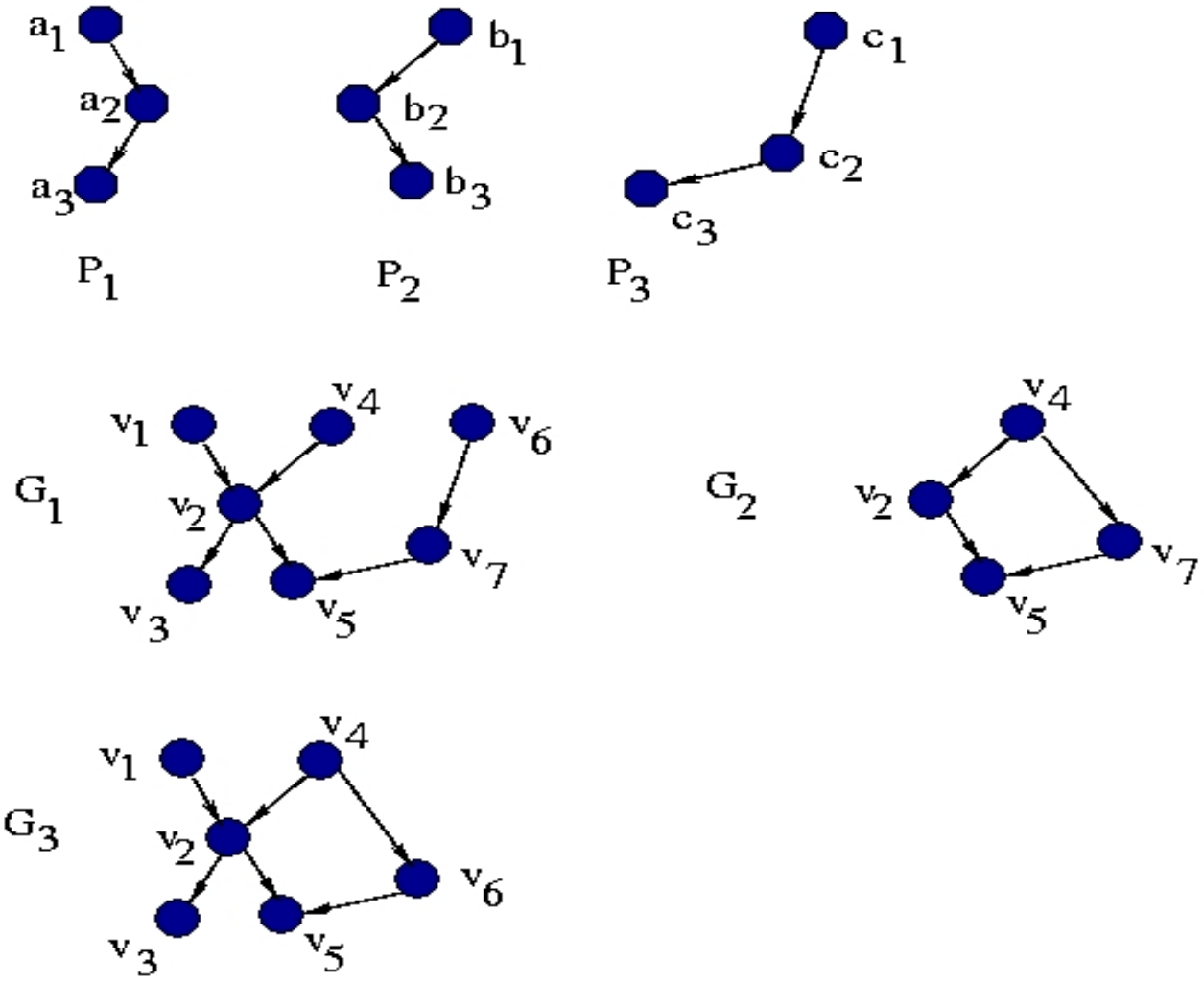
G_1			
		P_2	P_3
v1	v1		
v2	v2	b2	
v3	v3		
v4	v4	**b1**	
v5	v5	**b3**	**c3**
v6	v6		**c1**
v7	v7		c2

Splice G_1 with G_2

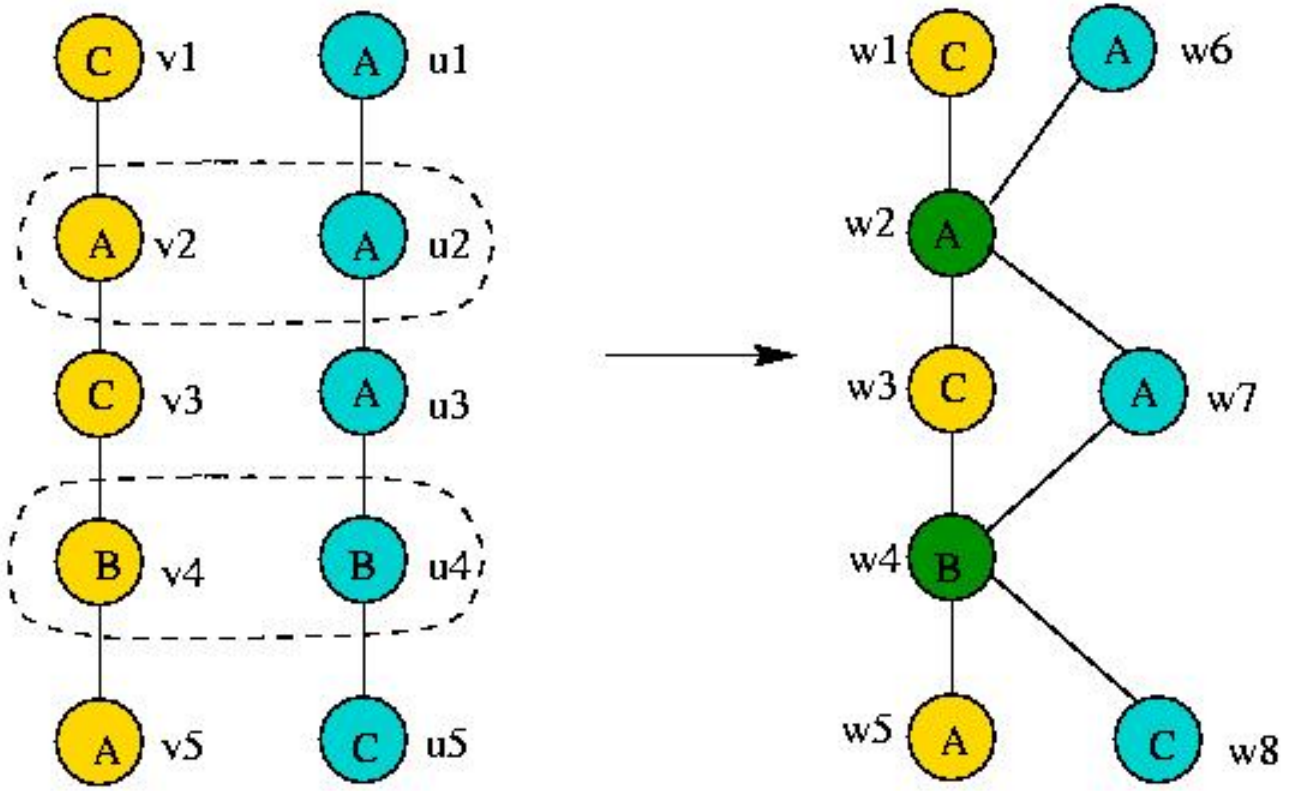
G_2		
	P_2	P_3
v2	**b1**	**c1**
v4	b2	
v5	**b3**	**c3**
v7		c2



G_3			
	P_1	P_2	P_3
v1	a1		
v2	a2	b2	
v3	a3		
v4		**b1**	**c1**
v5		**b3**	**c3**
v6			c2


a1
a2
a3
P1
b1
b2
b3
P2
c1
c2
c3
P3
G1
v1
v2
v3
v4
v5
v6
v7
G2
v2
v4
v5
v7
G3
v1
v2
v3
v4
v5
v6

We join only nodes that have the same labels!



1. Find all frequent edges.
2. Find frequent paths by adding one edge at a time
   (not all nodes are suitable for this!)
3. Find all frequent 2-path graphs by exhaustive joining.
4. Set k=2.
5. While frequent k-path graphs exist:
   a) Perform sum operation on pairs of frequent k-path graphs where applicable.
   b) Perform splice operation on generated (k+1)-path candidates To get additional (k+1)-path candidates.
   c) Compute support for (k+1)-path candidates.
   d) Eliminate non-frequent candidates and set k:=k+1.
   e) Go to 5.

on the size of the database (like any Apriori alg.)

Difficult tasks: (NP hard)

1. Support computation that consists of:
   a. Finding all instances of a frequent pattern in the database. (sub-graph isomorphism)
   b. Computing MIS (maximum independent set size) of an instance graph.

Relatively easy tasks:

1. Candidate set generation:
   polynomial on the size of frequent set from previous iteration,
2. Elimination of isomorphic candidate patterns:
   graph isomorphism computation is at worst exponential on the size of a pattern, not the database.

Additional Approaches for Single Graph Setting

- BFS Approach
  - hSiGram
- DFS Approach
  - vSiGram
- Both use approximations of the MIS measure

M. Kuramochi and G. Karypis

Finding Frequent Patterns in a Large Sparse Graph

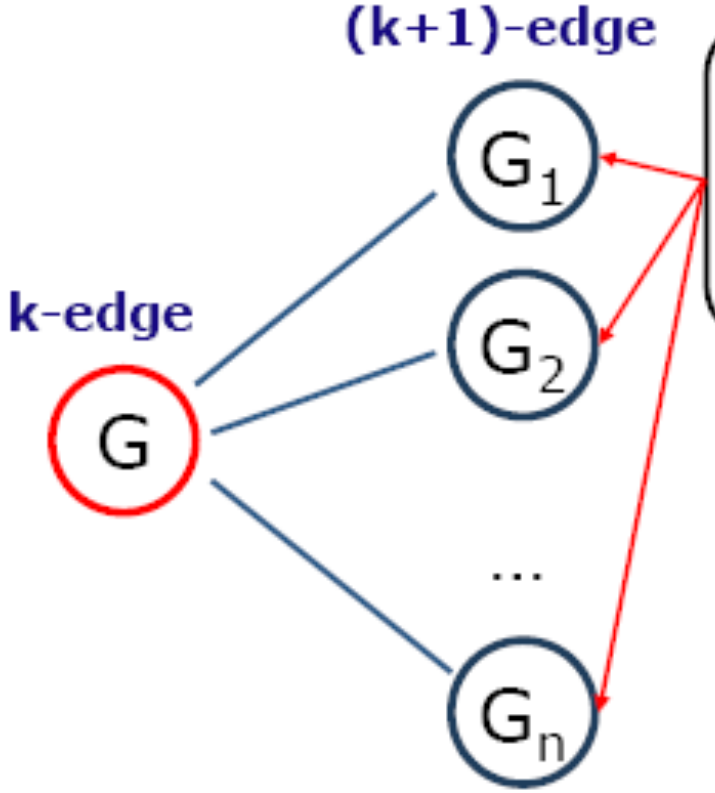
In Proc. Of SIAM 2004.

Closed Frequent Graphs

- Motivation: Handling graph pattern explosion problem
- Closed frequent graph
  - A frequent graph G is *closed* if there exists no supergraph of G that carries the same support as G
- If some of G's subgraphs have the same support, it is unnecessary to output these subgraphs (nonclosed graphs)
- *Lossless compression:* still ensures that the mining result is complete

CLOSEGRAPH (Yan and Han, KDD'03)

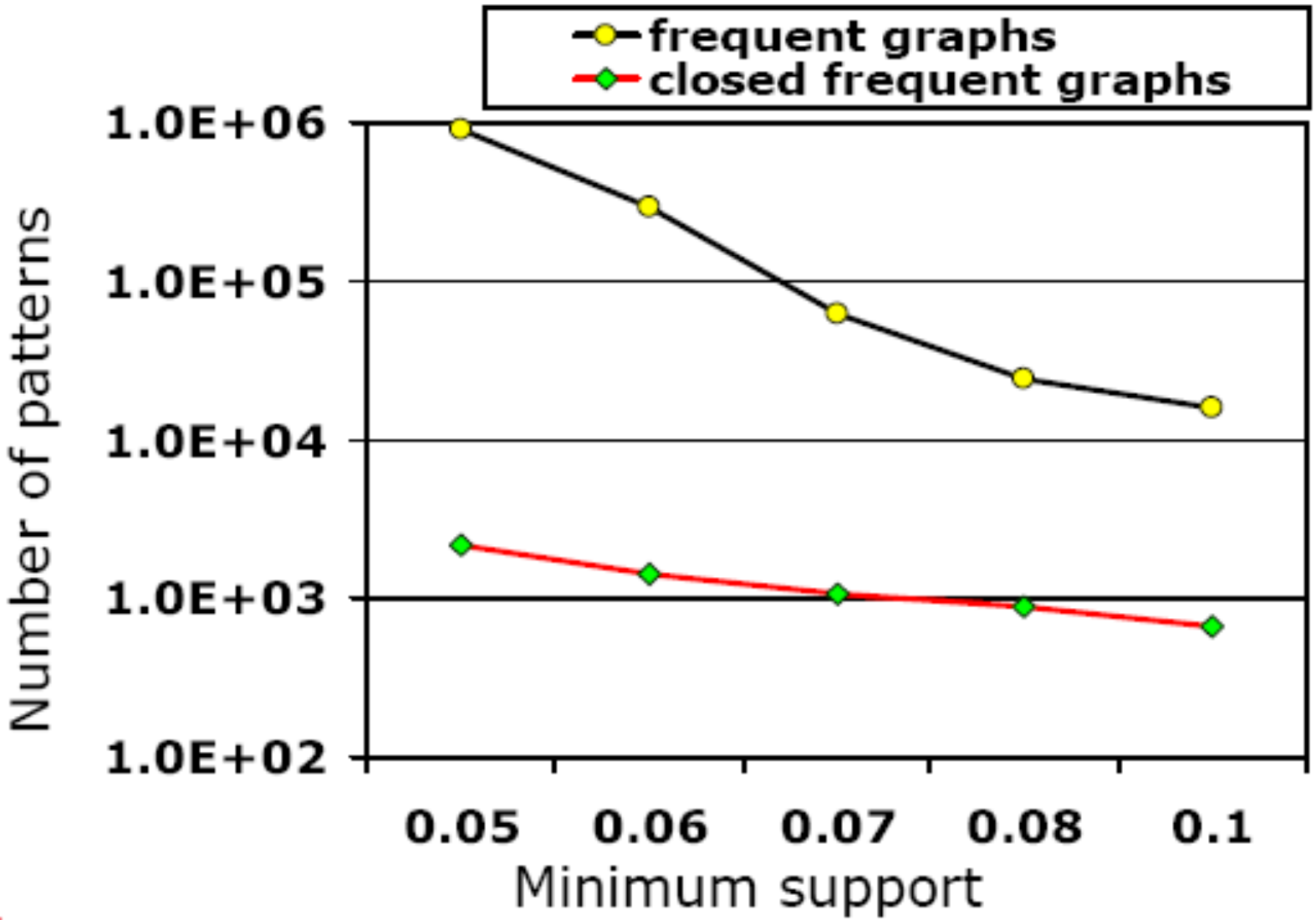
A Pattern-Growth Approach



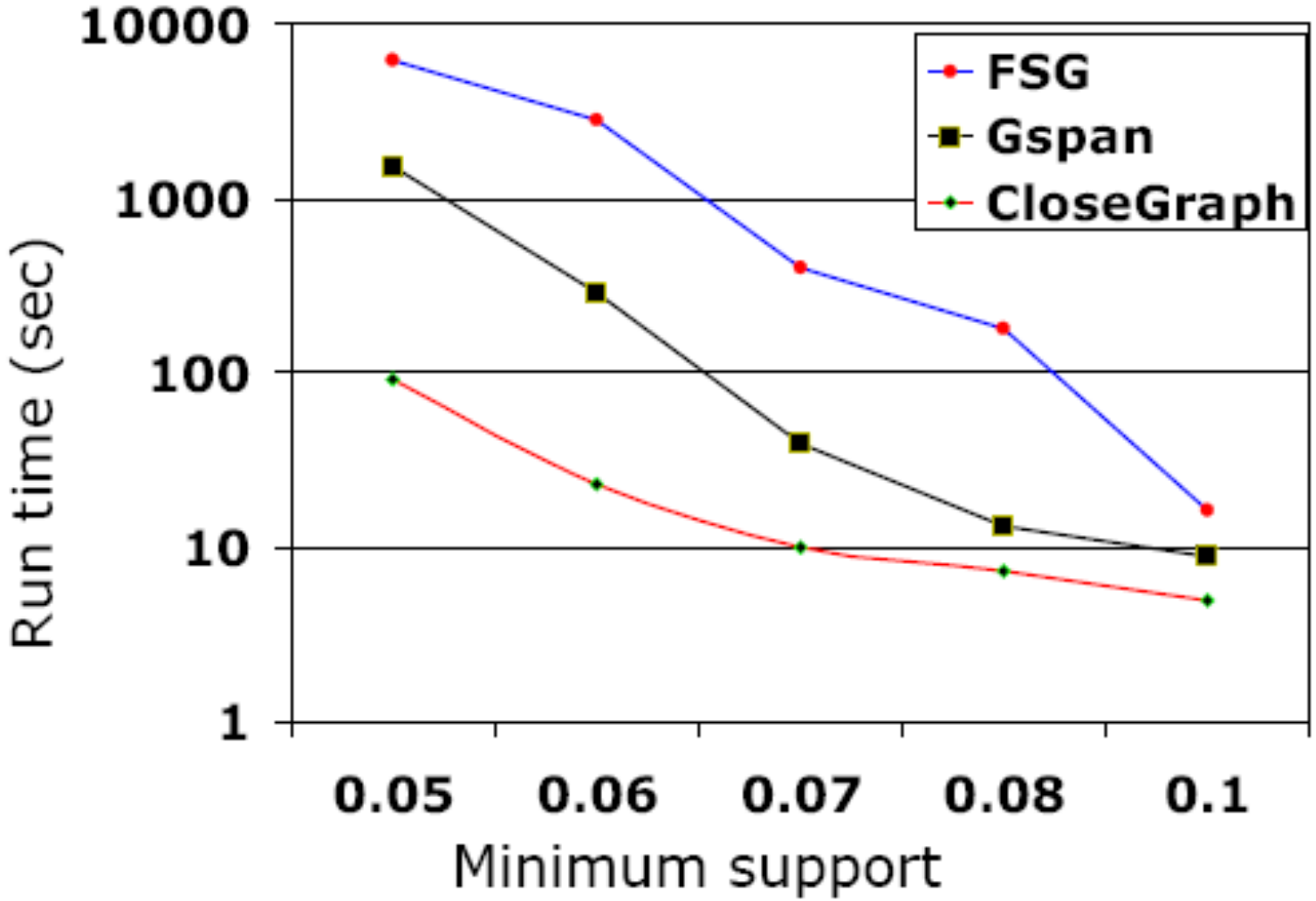
At what condition, can we stop searching their children i.e., early termination?

If G and G' are frequent, G is a subgraph of G'. If **in any part of graphs in the dataset where G occurs, G' also occurs**, then we need not grow G, since none of G's children will be closed except those of G'.

Number of Patterns: Frequent vs. Closed



Runtime: Frequent vs. Closed



Conclusions

- Data Mining field proved its practicality during its short lifetime with effective DM algorithms.
- Many applications in Databases, Chemistry&Biology, Networks, etc.
- Both Transaction and Single graph settings are important
- Graph Mining is:
  - Dealing with designing effective algorithms for mining graph datasets.
  - Facing many *hardness* problems on the way.
  - Fast growing field with many possibilities of evolving unseen before.
- As more and more information is stored in complicated structures, we need to develop new set of algorithms for Graph Data Mining.

Some References

[1] T. Washio A. Inokuchi and H.~Motoda, ***An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data***, Proceedings of the 4th PKDD'00, 2000, pages 13-23.

[2] M. Kuramochi and G. Karypis, ***An Efficient Algorithm for Discovering Frequent Subgraphs***, Tech. report, Department of Computer Science/Army HPC Research Center, 2002.

[3] N. Vanetik, E.Gudes, and S. E. Shimony, ***Computing Frequent Graph Patterns from Semistructured Data***, Proceedings of the 2002 IEEE ICDM'02

[4] Y. Xifeng and H. Jiawei, ***gspan: Graph-Based Substructure Pattern Mining***, Tech. report, University of Illinois at Urbana-Champaign, 2002.

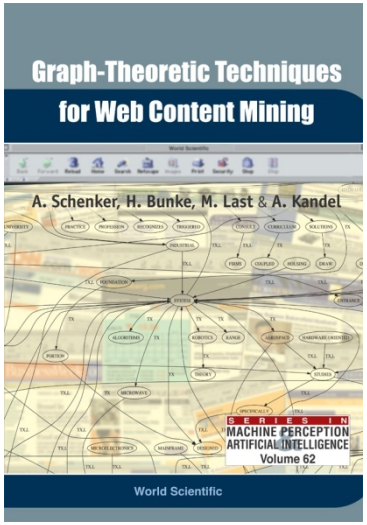
[5] W. Wang J. Huan and J. Prins, ***Efficient Mining of Frequent Subgraphs in the Presence of Isomorphism***, Proceedings of the 3rd IEEE ICDM'03 p.~549.

[6] Moti Cohen, Ehud Gudes, ***Diagonally Subgraphs Pattern Mining***. DMKD 2004, pages 51-58, 2004

[7] D. J. Cook and L. B. Holder, ***Graph-Dased Data Mining***, Tech. report, Department of CS Engineering, 1998.

Documents Classification:

Alternative Representation of Multilingual Web Documents:The Graph-Based Model



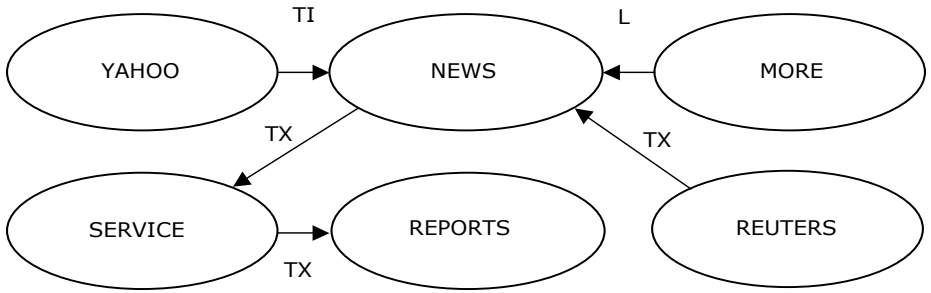
Introduced in A. Schenker, H. Bunke, M. Last, A. Kandel, *Graph-Theoretic Techniques for Web Content Mining*, World Scientific, 2005

The Graph-Based Model of Web Documents

- Basic ideas:
  - One node for each unique term
  - If word B follows word A, there is an edge from A to B
    - In the presence of terminating punctuation marks (periods, question marks, and exclamation points) no edge is created between two words
  - Graph size is limited by including only the most frequent terms
  - Several variations for node and edge labeling (see the next slides)
- Pre-processing steps
  - Stop words are removed
  - Lemmatization
    - Alternate forms of the same term (singular/plural, past/present/future tense, etc.) are conflated to the most frequently occurring form

The *Standard* Representation

- Edges are labeled according to the document section where the words are followed by each other
  - *Title (TI)* contains the text related to the document's title and any provided keywords (meta-data);
  - *Link (L)* is the "anchor text" that appears in clickable hyper-links on the document;
  - *Text (TX)* comprises any of the visible text in the document (this includes anchor text but not title and keyword text)



Graph Based Document Representation – Detailed Example

Source: www.cnn.com, May 24, 2005

Iraq bomb: Four dead, 110 wounded

A car bomb has exploded outside a popular Baghdad restaurant, killing three Iraqis and wounding more than 110 others, police officials said. Earlier an aide to the office of Iraqi Prime Minister Ibrahim al-Jaafari and his driver were killed in a drive-by shooting.

FULL STORY

Graph Based Document Representation - Parsing

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//E
<!-- saved from url=(0023)http://edition.cnn.com/ -->
<HTML lang=en><HEAD><TITLE>CNN.com International</TITLE>
<META http-equiv=content-type content="text/html; charset=iso-8859-1">
<META http-equiv=refresh content=1800><LINK href="/" rel=Start><LINK


<DIV class=cnnSectionT1
style="PADDING-RIGHT: 6px; PADDING-LEFT: 6px; PADDING-BOTTOM: 1px; PADDING-TOP: 3px">
<H2><A style="COLOR: #000"
href="http://edition.cnn.com/2005/WORLD/meast/05/23/iraq.main/index.html">Iraq
bomb: Four dead, 110 wounded</A></H2>
<P>A car bomb has exploded outside a popular Baghdad restaurant, killing
three Iraqis and wounding more than 110 others, police officials said.
Earlier an aide to the office of Iraqi Prime Minister Ibrahim al-Jaafari
and his driver were killed in a drive-by shooting.</P>
<P><A class=cnnt1link
href="http://edition.cnn.com/2005/WORLD/meast/05/23/iraq
STORY</A></P>
```

title

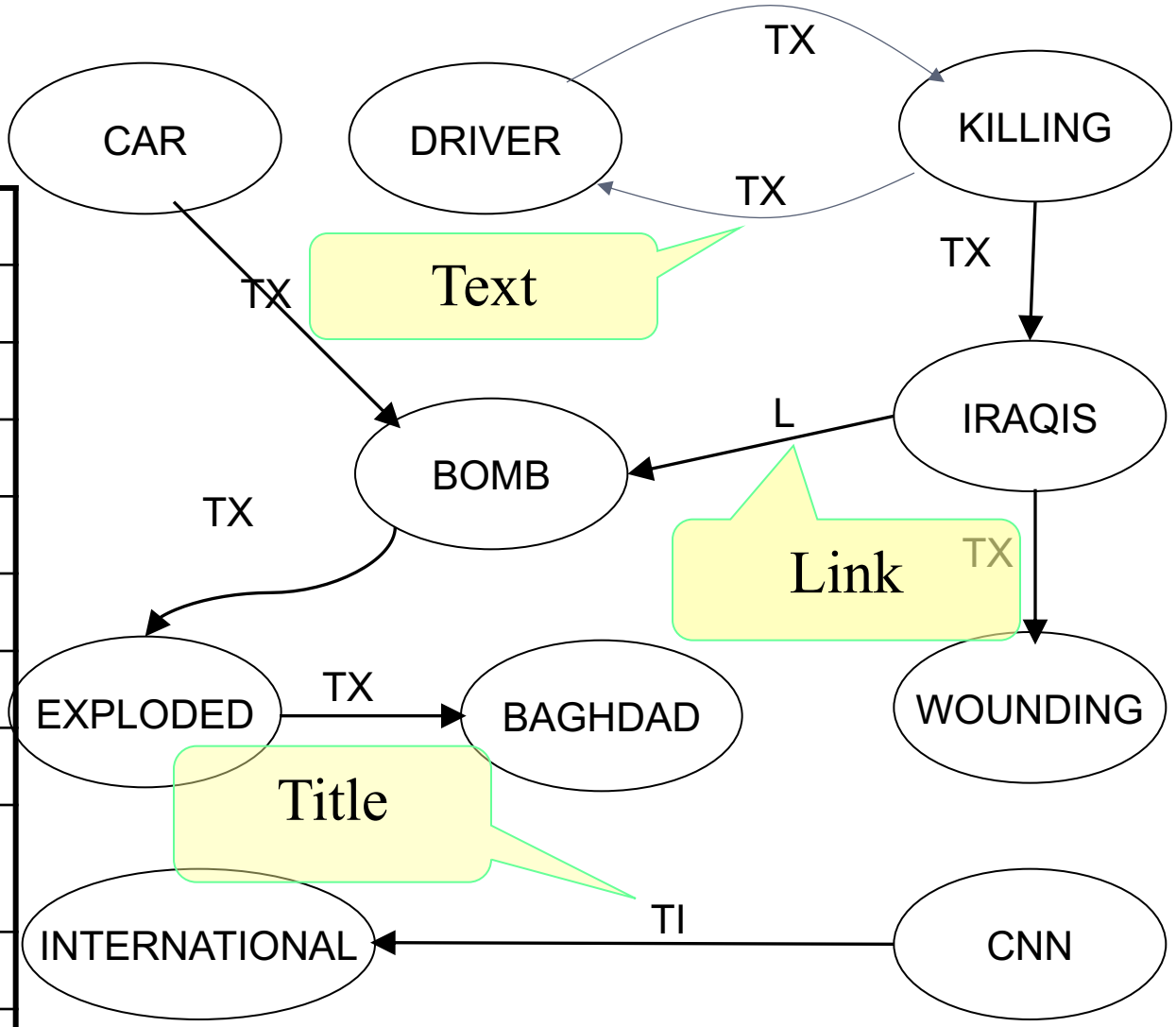
link

text

Standard Graph Based Document Representation

Ten most frequent terms are used

Word	Frequency
Iraqis	3
Killing	2
Bomb	2
Wounding	2
Driver	2
Exploded	1
Baghdad	1
International	1
CNN	1
Car	1



Classification using graphs

- Basic idea:
  - Mine the frequent sub-graphs, call them terms
  - Use TFIDF for assigning the most characteristic terms to documents
  - Use Clustering and K-nearest neighbors classification

Subgraph Extraction

- Input
  - ***G*** – training set of directed, unique nodes graphs
  - CR_{min} - Minimum Classification Rate
- Output
  - Set of classification-relevant sub-graphs
- Process:
  - For each class find subgraphs $\boldsymbol{CR} > \boldsymbol{CR}_{\text{min}}$
  - Combine all sub-graphs into one set
- Basic Assumption
  - **Classification-Relevant Sub-Graphs** are more frequent in a specific category than in other categories

Computing the Classification Rate

- Subgraph Classification Rate:

$$CR\left(g'_k\left(c_i\right)\right) = SCF\left(g'_k\left(c_i\right)\right) \times ISF\left(g'_k\left(c_i\right)\right)$$

- *SCF* $(g'_k(c_i))$ - Subgraph Class Frequency of subgraph g'_k in category c_i
- *ISF* $(g'_k(c_i))$ - Inverse Subgraph Frequency of subgraph g'_k in category c_i
- **Classification Relevant Feature** is a feature that best explains a specific category, or frequent in this category more than in all others

k-Nearest Neighbors with Graphs

Accuracy vs. Graph Size