MINING AND SEARCHING GRAPHS AND STRUCTURES

Jiawei Han Xifeng Yan

Department of Computer Science

University of Illinois at Urbana-Champaign

Philip S. Yu

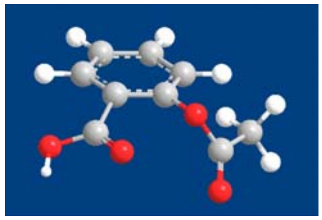
IBM T. J. Watson Research Center

http://ews.uiuc.edu/~xyan/tutorial/kdd06_graph.htm

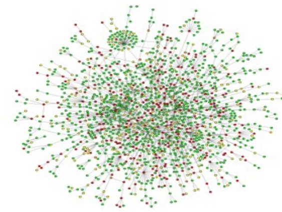
Outline

- Scalable pattern mining in graph data sets
  - Frequent subgraph pattern mining
  - Constraint-based graph pattern mining
  - Pattern summarization / selection
  - Graph clustering, classification, and compression
- Searching graph databases
  - Graph indexing methods
  - Substructure similarity search
  - Search with constraints
- Application and exploration with graph mining
  - Biological and social network analysis
  - Mining software systems: bug isolation & performance tuning
- Conclusions

Graph, Graph, Everywhere

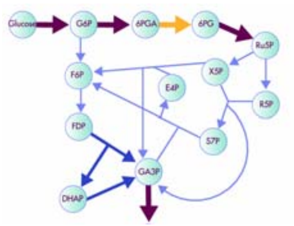


Aspirin

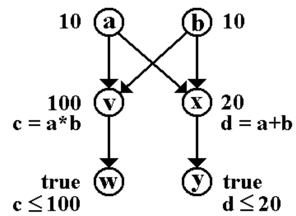


Yeast Protein Interaction Network

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

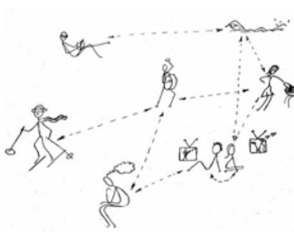


Metabolic Network



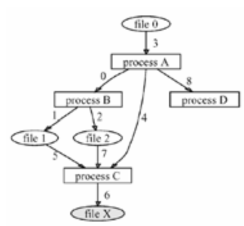
Dependency Graph

Graph, Graph, Everywhere (cont.)

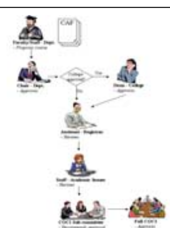


Social Network

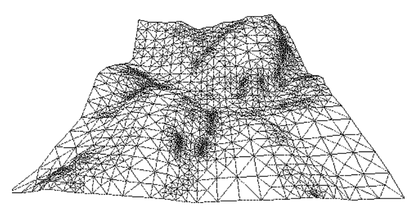
from Adamic etc. A social network caught in the web (2003)



Event Log Graph



Workflow



Mesh

Motivation

- Graph is ubiquitous
  - Model complex data
- 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 graph algorithms
  - Many problems are of high complexity
  - “NP hard” doesn’t shadow their values

Outline

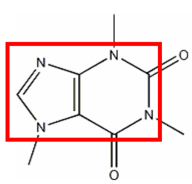
- Scalable pattern mining in graph data sets
  - Frequent subgraph pattern mining
  - Constraint-based graph pattern mining
  - Pattern summarization / selection
  - Graph clustering, classification, and compression
- Searching graph databases
  - Graph indexing methods
  - Substructure similarity search
  - Search with constraints
- Application and exploration with graph mining
  - Biological and social network analysis
  - Mining software systems: bug isolation & performance tuning
- Conclusions

Graph Pattern Mining

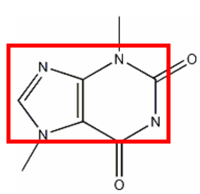
- Frequent subgraphs
  - A (sub)graph is ***frequent*** if its *support* (occurrence frequency) in a given dataset is no less than a *minimum support* threshold
- Applications of graph pattern mining
  - Mining biochemical structures
  - Mining biological conserved subnetworks
  - Program control flow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

Example: Frequent Subgraphs

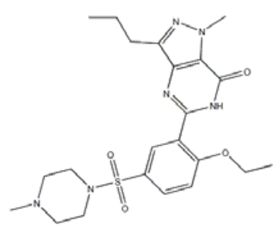
CHEMICAL COMPOUNDS



(a) caffeine



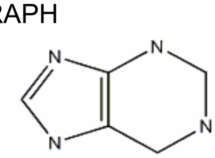
(b) diurobromine



(c) viagra

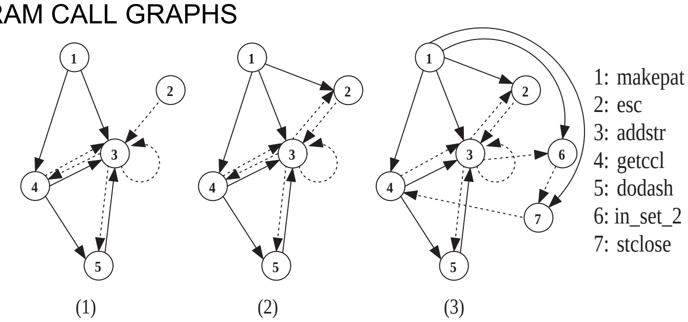
. . .

FREQUENT SUBGRAPH

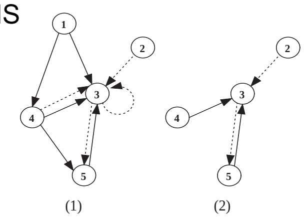


Example (cont.)

PROGRAM CALL GRAPHS



FREQUENT SUBGRAPHS
(MIN SUPPORT IS 2)

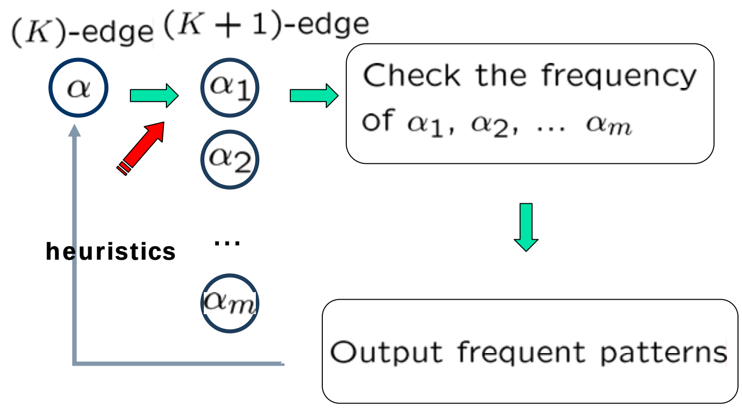


Graph Mining Algorithms

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

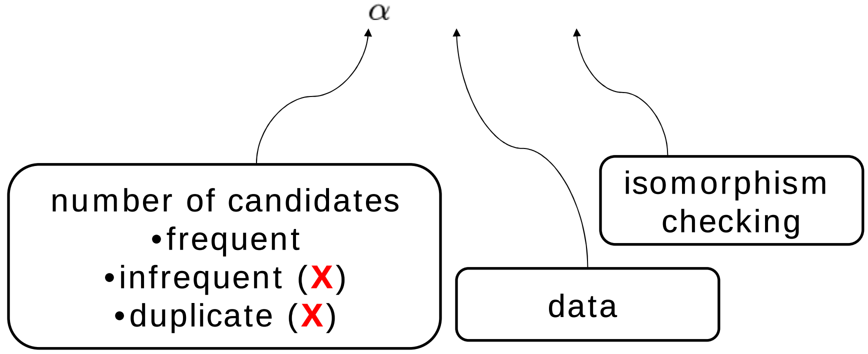
Apriori Property

If a graph is frequent, all of its subgraphs are frequent.



Cost Analysis

$$T_{total} \propto \sum_{\alpha} |D_\alpha| \times T_\alpha^{iso}$$



SUBDUE (Holder et al. KDD'94)

- Start with single vertices
- Expand best substructures with a new edge
- Limit the number of best substructures
  - Substructures are evaluated based on their ability to compress input graphs
  - Using minimum description length (DL)
  - Best substructure *S* in graph *G* minimizes: DL(S) + DL(G\S)
- Terminate until no new substructure is discovered

WARMR (Dehaspe et al. KDD'98)

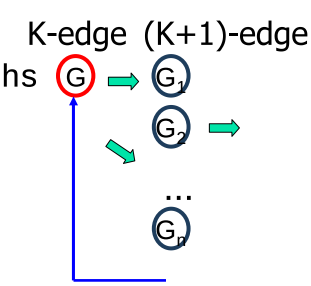
- Graphs are represented by Datalog facts
  - *atomel(C, A1, c), bond (C, A1, A2, BT), atomel(C, A2, c) : a carbon atom bound to a carbon atom with bond type BT*
- WARMR: the first general purpose ILP system
- Level-wise search
- Simulate Apriori for frequent pattern discovery

Frequent Subgraph Mining Approaches

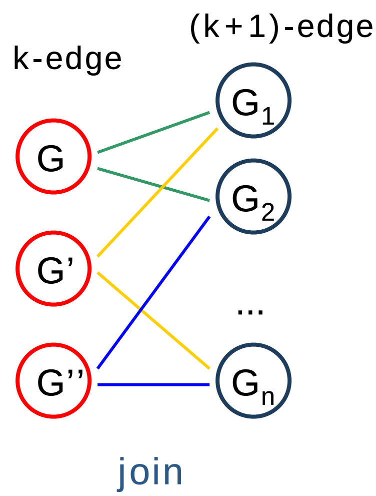
- Apriori-based approach
  - AGM/AcGM: Inokuchi, et al. (PKDD'00)
  - FSG: Kuramochi and Karypis (ICDM'01)
  - PATH#: Vanetik and Gudes (ICDM'02, ICDM'04)
  - FFSM: Huan, et al. (ICDM'03)
- Pattern growth approach
  - MoFa: Borgelt and Berthold (ICDM'02)
  - gSpan: Yan and Han (ICDM'02)
  - Gaston: Nijssen and Kok (KDD'04)

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
- Discovery order of patterns
  - path → tree → graph

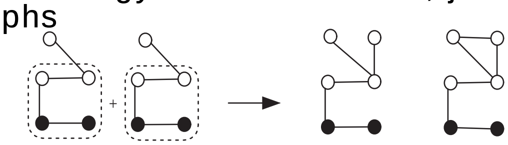


Apriori-Based Approach

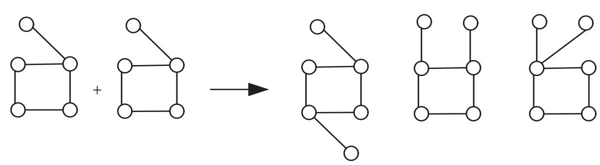


Apriori-Based, Breadth-First Search

- Methodology: breadth-search, joining two graphs



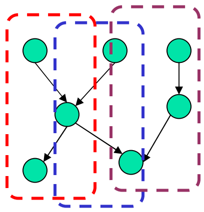
- AGM (Inokuchi, et al. PKDD'00)
  - generates new graphs with one more node



- FSG (Kuramochi and Karypis ICDM'01)
  - generates new graphs with one more edge

PATH (Vanetik and Gudes ICDM'02, '04)

- Apriori-based approach
- Building blocks: edge-disjoint path



A graph with 3 edge-disjoint paths

- construct frequent paths
- construct frequent graphs with 2 edge-disjoint paths
- construct graphs with k+1 edge-disjoint paths from graphs with k edge-disjoint paths
- repeat

FFSM (Huan, et al. ICDM'03)

X			
2	Y		
	1	X	
	1		Z

\+

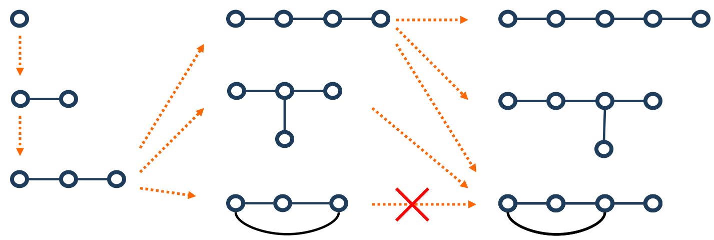
X			
2	Y		
	1	X	
		1	Z

→

X			
2	Y		
	1	X	
	1	1	Z

- Represent graphs using canonical adjacency matrix (CAM)
- Join two CAMs or extend a CAM to generate a new graph
- Store the embeddings of CAMs
  - All of the embeddings of a pattern in the database
  - Can derive the embeddings of newly generated CAMs

Pattern Growth Method



- detect duplicates

MoFa (ICDM'02)

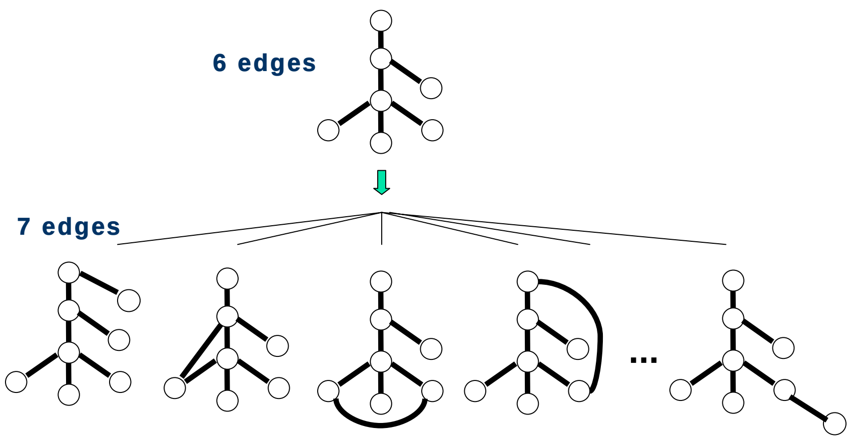
- avoid duplicates

❑ gSpan (ICDM'02)

MoFa (Borgelt and Berthold ICDM'02)

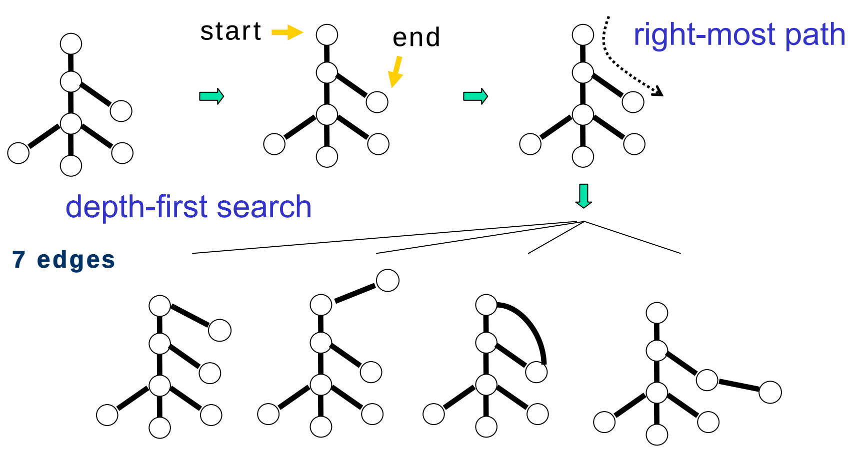
- Extend graphs by adding a new edge
- Store embeddings of discovered frequent graphs
  - Fast support calculation
  - Also used in other later developed algorithms such as FFSM and GASTON
  - Expensive Memory usage
- Local structural pruning

Free Extension



22 new graphs

Right-Most Extension (Yan and Han ICDM'02)



4 new graphs

GSPAN (Yan and Han ICDM'02)

Theorem: Completeness

The Enumeration of Graphs Using Right-Most Extension is COMPLETE.

GASTON (Nijssen and Kok KDD'04)

- Extend graphs directly
- Store embeddings
- Separate the discovery of different types of graphs
  - path → tree → graph
  - Simple structures are easier to mine and duplication detection is much simpler

Graph Pattern Explosion Problem

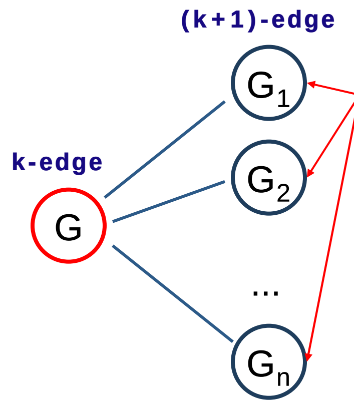
- If a graph is frequent, all of its subgraphs are frequent ─ **the Apriori property**
- An **n**-edge frequent graph may have 2^n subgraphs
- Among **423** chemical compounds which are confirmed to be active in an AIDS antiviral screen dataset, there are around **1,000,000** frequent graph patterns if the minimum support is 5%

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)

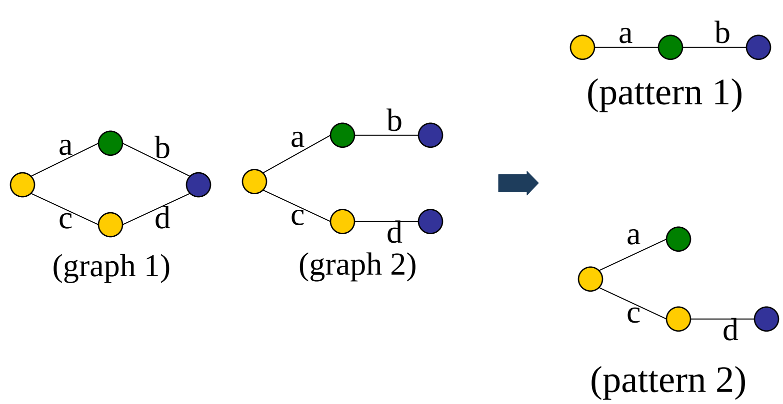
A Pattern-Growth Approach



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′.

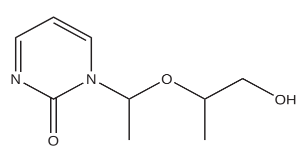
Handling Tricky Exception Cases



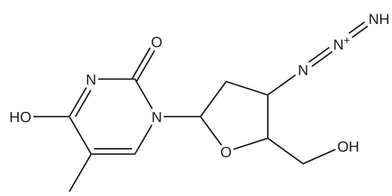
Experimental Result

- The AIDS antiviral screen compound dataset from NCI/NIH
- The dataset contains 43,905 chemical compounds
- Among these 43,905 compounds, 423 of them belong to CA, 1081 are of CM, and the rest is in class CI

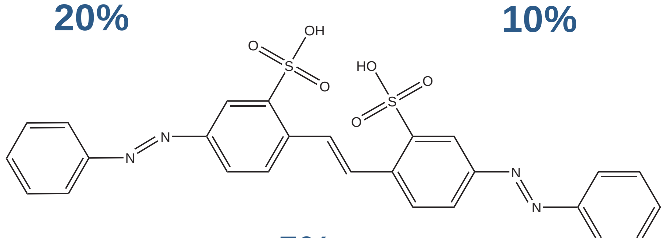
Discovered Patterns



20%

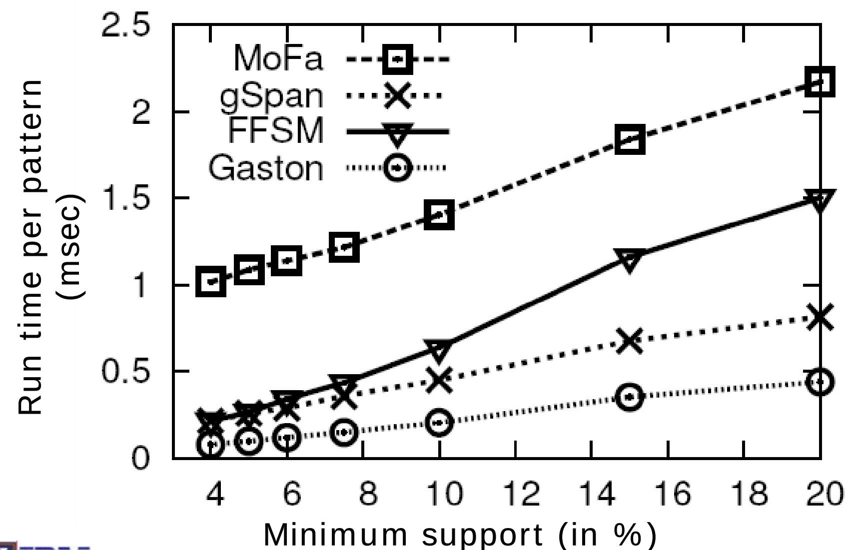


10%

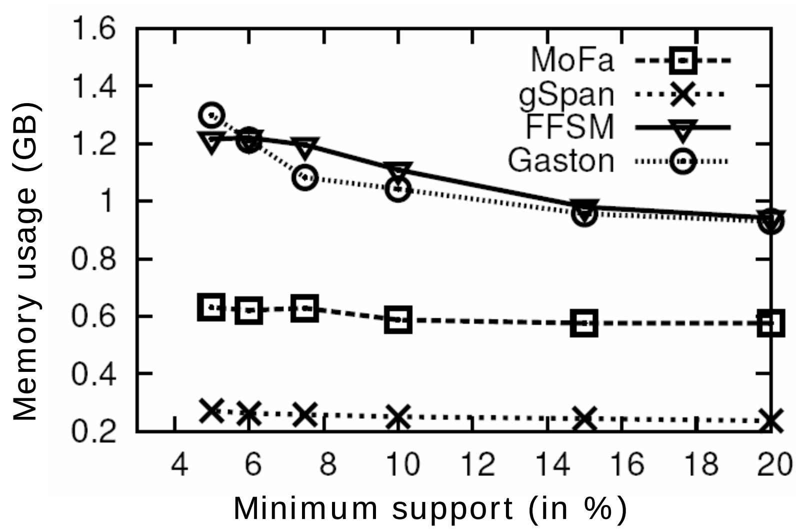


5%

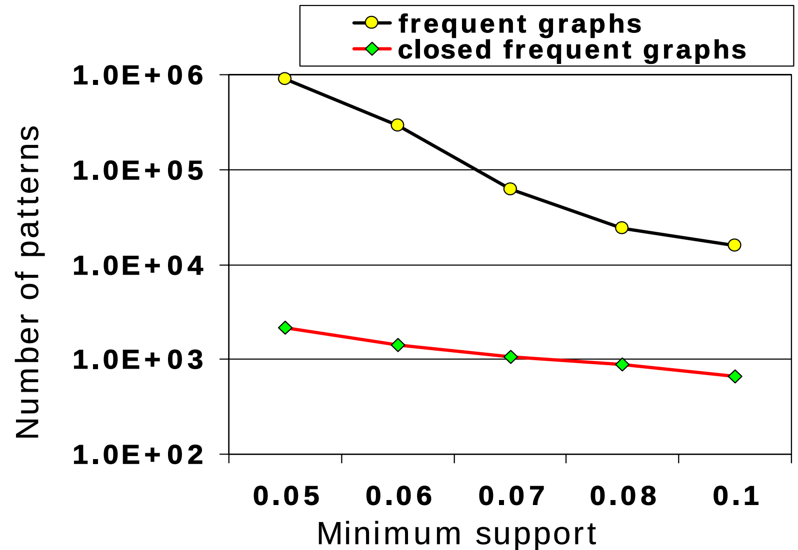
Performance: Run Time



Performance: Memory Usage

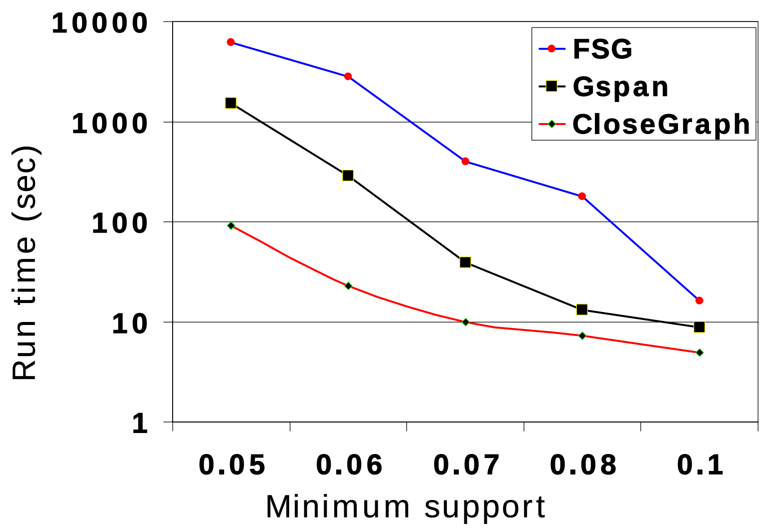


Number of Patterns: Frequent vs. Closed



 Mining and Searching Graphs and Structures 35

Runtime: Frequent vs. Closed



 Mining and Searching Graphs and Structures 36

Outline

- Scalable pattern mining in graph data sets
  - Frequent subgraph pattern mining
  - Constraint-based graph pattern mining
  - Pattern summarization / selection
  - Graph clustering, classification, and compression
- Searching graph databases
  - Graph indexing methods
  - Substructure similarity search
  - Search with constraints
- Application and exploration with graph mining
  - Biological and social network analysis
  - Mining software systems: bug isolation & performance tuning
- Conclusions

Graph Constraints

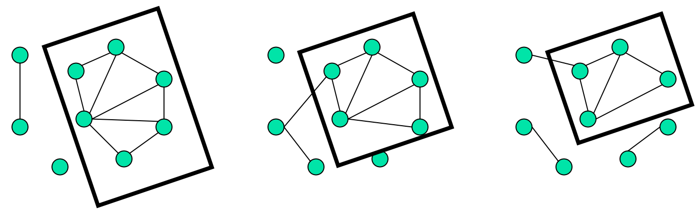
A constraint C is a boolean predicate, $C : P \rightarrow \{0, 1\}$, which maps a pattern α to a Boolean value. A pattern α satisfies constraint C if $C(\alpha) = 1$.

graph constraints

- Degree
- Size
- Density
- Density ratio
- Diameter
- Edge connectivity
- Vertex connectivity
- Aggregation (min, max, avg)

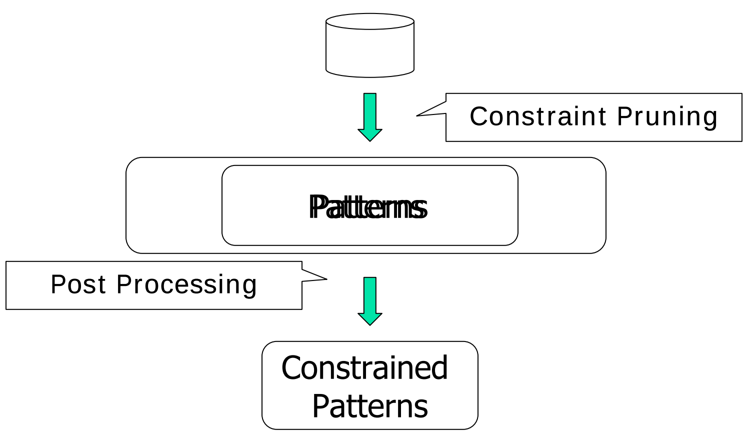
Constraint-Based Graph Pattern Mining

- Highly connected subgraphs in a large graph usually are not artifacts (group, functionality)

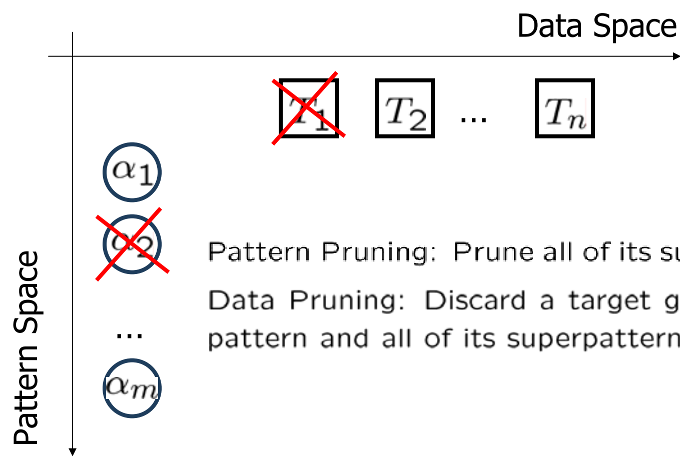


- Recurrent patterns discovered in multiple graphs are more robust than the patterns mined from a single graph

Push Constraints Deep



Pruning Patterns vs Data

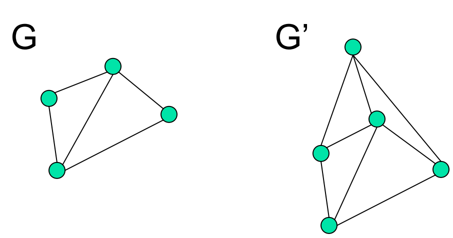


Pattern Pruning: Prune all of its superpatterns

Data Pruning: Discard a target graph for the pattern and all of its superpatterns

No Downward Closure Property

Given two graphs G and G', if G is a subgraph of G', it does not imply that the connectivity of G is less than that of G', and vice versa.



Pattern/Data Space Pruning

- Pattern space pruning
  - Strong P-antimonotonicity
  - Weak P-antimonotonicity
- Data space pruning
  - Pattern-separable D-antimonotonicity
  - Pattern-inseparable D-antimonotonicity

Antimonotonicity Summary

Constraint	strong P-antimonotone	weak P-antimonotone	pattern-separable D-antimonotone	pattern-inseparable D-antimonotone
$Min\_Degree(P) \geq \delta$	No	No	No	Yes
$Min\_Degree(P) \leq \delta$	No	Yes	No	Yes
$Max\_Degree(P) \geq \delta$	No	No	Yes	Yes
$Max\_Degree(P) \leq \delta$	Yes	Yes	No	Yes
$Density\_Ratio(P) \geq \delta$	No	Yes	No	Yes
$Density\_Ratio(P) \leq \delta$	No	Yes	No	Yes
$Density(P) \geq \delta$	No	No	No	Yes
$Density(P) \leq \delta$	No	Yes	No	Yes
$Size(P) \geq \delta$	No	Yes	Yes	Yes
$Size(P) \leq \delta$	Yes	Yes	No	Yes
$Diameter(P) \geq \delta$	No	Yes	No	Yes
$Diameter(P) \leq \delta$	No	No	No	Yes
$EdgeConnectivity(P) \geq \delta$	No	No	No	Yes
$EdgeConnectivity(P) \leq \delta$	No	Yes	No	Yes
$VertexConnectivity(P) \geq \delta$	No	No	No	Yes
$VertexConnectivity(P) \leq \delta$	No	Yes	No	Yes
P contains a benzene ring	No	Yes	Yes	Yes
P does not contain a benzene ring	Yes	Yes	No	Yes

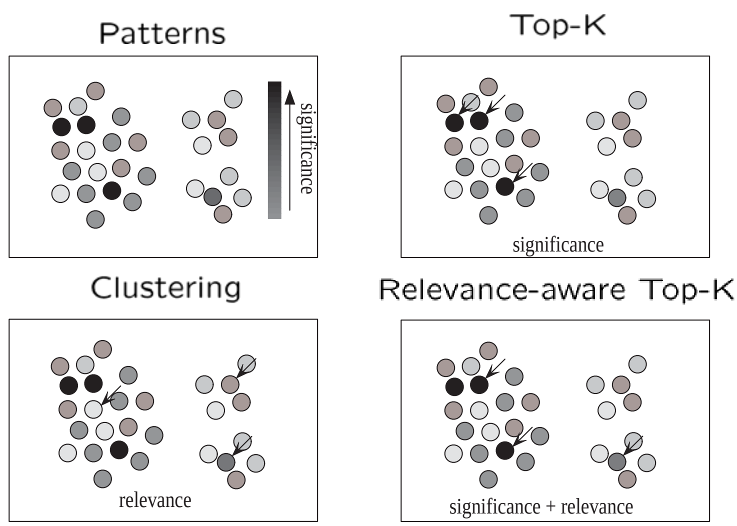
Outline

- Scalable pattern mining in graph data sets
  - Frequent subgraph pattern mining
  - Constraint-based graph pattern mining
  - Pattern summarization / selection
  - Graph clustering, classification, and compression
- Searching graph databases
  - Graph indexing methods
  - Substructure similarity search
  - Search with constraints
- Application and exploration with graph mining
  - Biological and social network analysis
  - Mining software systems: bug isolation & performance tuning
- Conclusions

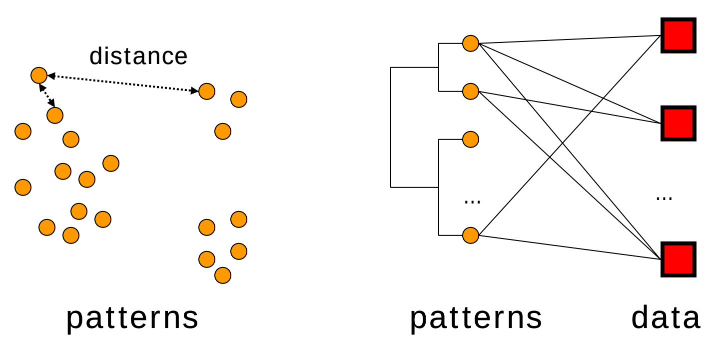
Pattern Summarization

- Too many patterns may not lead to more explicit knowledge
- It can confuse users as well as further discovery (e.g., clustering, classification, indexing, etc.)
- A small set of "representative" patterns that preserve most of the information

Summarization Scenarios (KDD'07)



Pattern Distance



measure 1: pattern based
- pattern containment
- pattern similarity

measure 2: data based
- data similarity

Pattern Containment (Afrati et al. KDD'04)

Pattern Based

Given a pattern set $\mathcal{F}$, find a subset S ($|S| = k$) that optimizes

$$\frac{|\cup_{\alpha \in S} \mathcal{P}(\alpha)|}{|\mathcal{F}|}$$

Relaxed Pattern Based

Given a pattern set $\mathcal{F}$, find a set S ($|S| = k$) that optimizes

$$f_{+}(S) = \frac{|\cup_{\alpha \in S} \mathcal{P}(\alpha) \backslash D|}{|\cup_{\alpha \in S} \mathcal{P}(\alpha) \cap D|}$$

Data Similarity (VLDB'06, KDD'06)

Set Based

$$sim(D_\alpha, D_\beta) \sim \frac{|D_\alpha \cap D_\beta|}{|D_\alpha \cup D_\beta|}$$

jaccard distance

Model Based

$$M_\alpha \sim D_\alpha, \;\; M_\beta \sim D_\beta$$

$$sim(D_\alpha, D_\beta) \sim sim(M_\alpha, M_\beta)$$

Outline

- Scalable pattern mining in graph data sets
  - Frequent subgraph pattern mining
  - Constraint-based graph pattern mining
  - Pattern summarization / selection
  - Graph clustering, classification, and compression
- Searching graph databases
  - Graph indexing methods
  - Substructure similarity search
  - Search with constraints
- Application and exploration with graph mining
  - Biological and social network analysis
  - Mining software systems: bug isolation & performance tuning
- Conclusions

Graph Clustering

- Graph similarity measure
  - Feature-based similarity measure
    - Each graph is represented as a feature vector
    - The similarity is defined by the distance of their corresponding vectors
    - Frequent subgraphs can be used as features
  - Structure-based similarity measure
    - Maximal common subgraph
    - Graph edit distance: insertion, deletion, and relabel
    - Graph alignment distance

Graph Classification

- Local structure based approach
  - Local structures in a graph, e.g., neighbors surrounding a vertex, paths with fixed length
- Graph pattern based approach
  - Subgraph patterns from domain knowledge
  - Subgraph patterns from data mining
- Kernel-based approach
  - Random walk (Gärtner '02, Kashima et al. '02, ICML'03, Mahé et al. ICML'04)
  - Optimal local assignment (Fröhlich et al. ICML'05)
- Boosting (Kudo et al. NIPS'04)

Graph Pattern Based Classification

- Subgraph patterns from domain knowledge
  - Molecular descriptors
- Subgraph patterns from data mining
- General idea
  - Each graph is represented as a feature vector **x** = $\{x_1, x_2, \ldots, x_n\}$, where x_i is the frequency of the i-th pattern in that graph
  - Each vector is associated with a class label
  - Classify these vectors in a vector space

Subgraph Patterns from Data Mining

- Sequence patterns (De Raedt and Kramer IJCAI'01)
- Frequent subgraphs (Deshpande et al, ICDM'03)
- Coherent frequent subgraphs (Huan et al. RECOMB'04)
  - A graph *G* is *coherent* if the mutual information between *G* and each of its own subgraphs is above some threshold

$$p(X_G = 1) = \text{frequency of } G$$

$$I(G, G') = \sum_{X_G, X_{G'}} p(X_G, X_{G'}) log \frac{p(X_G, X_{G'})}{p(X_G)p(X_{G'})}$$

- Closed frequent subgraphs (Liu et al. SDM'05)
- Acyclic Subgraphs (Wale and Karypis, technical report '06)

Kernel-based Classification

- Random walk
  - Marginalized Kernels (Gärtner '02, Kashima et al. '02, ICML'03, Mahé et al. ICML'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., $K_L(l_1, l_2) = \begin{cases} 1 & \text{if } l_1 = l_2, \\ 0 & otherwise. \end{cases}$

Kernel-based Classification

- Optimal local assignment (Fröhlich et al. ICML'05)

$$K(G,G') = \begin{cases} max_\pi \sum_{i=1}^{|V(G)|} \boxed{k(v_i, v'_{\pi_i})} & \text{if } |V(G)| \geq |V(G')|, \\ max_\pi \sum_{i=1}^{|V(G')|} k(v_{\pi_i}, v'_i) & otherwise. \end{cases}$$

can be extended to include neighborhood information e.g.,

$$k_{nei}(v,v') = k_{atom}(v,v') + \sum_{l=0}^{L} \lambda_l R_l(v,v')$$

where R_l could be an RBF-kernel to measure the similarity of neighborhoods of vertices v and v', λ_l is a damping parameter.

Boosting in Graph Classification

- Decision stumps (Kudo et al. NIPS'04)
  - 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

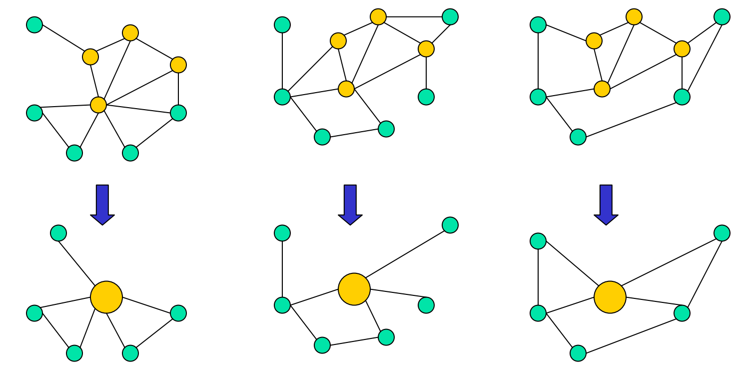
$$h_{<t,y>}(\mathbf{x}) = \begin{cases} y & \text{if } t \subseteq \mathbf{x}, \\ -y & otherwise. \end{cases}$$

  - Gain $gain(< t, y >) = \sum_{i=1}^{n} y_i h_{<t,y>}(\mathbf{x}_i)$
  - Applying boosting

$$gain(< t, y >) = \sum_{i=1}^{n} y_i \boxed{d_i} h_{<t,y>}(\mathbf{x}_i)$$

Graph Compression (Holder et al., KDD'94)

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

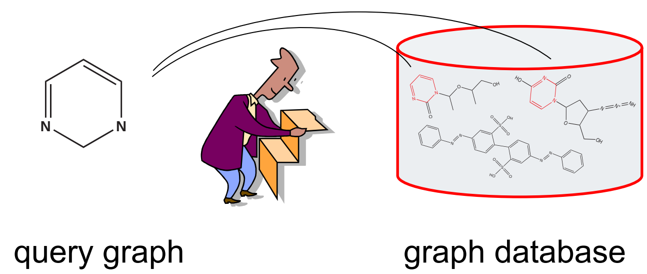


Outline

- Scalable pattern mining in graph data sets
  - Frequent subgraph pattern mining
  - Constraint-based graph pattern mining
  - Pattern summarization / selection
  - Graph clustering, classification, and compression
- Searching graph databases
  - Graph indexing methods
  - Substructure similarity search
  - Search with constraints
- Application and exploration with graph mining
  - Biological and social network analysis
  - Mining software systems: bug isolation & performance tuning
- Conclusions

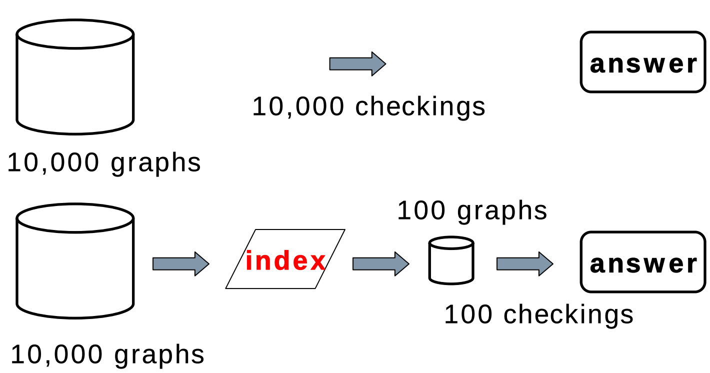
Graph Search

Find all of the graphs in a database that contain the query graph



Indexing Graphs

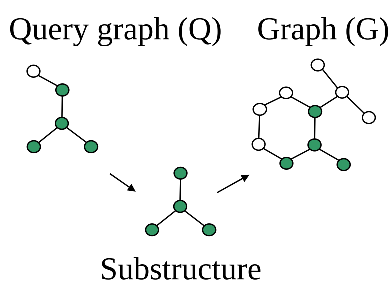
- Indexing is crucial



Scalability Issue

- Sequential scan
  - Disk I/Os
  - Subgraph isomorphism testing
- An indexing mechanism is needed
  - DayLight: Daylight.com (commercial)
  - GraphGrep: Dennis Shasha, et al. PODS'02
  - Grace: Srinath Srinivasa, et al. ICDE'03

Indexing Strategy



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

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

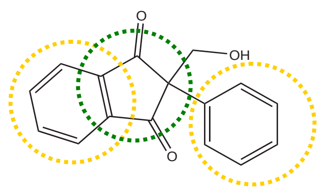
- Enumerate structures in the query graph
- Calculate the candidate graphs containing these structures
- Prune the false positive answers by performing subgraph isomorphism test

Feature-based Index

Question: What kind of substructures to index?

Options:

1. Node/edge labels
2. All of the substructures
3. Paths (Shasha et al. PODS'02)
4. Frequent graphs
5. Discriminative frequent graphs (Yan et al. SIGMOD'04)



Cost Analysis

QUERY RESPONSE TIME

$$T_{index} + \boxed{\left| C_q \right|} \times \left(T_{io} + T_{isomorphism\_testing} \right)$$

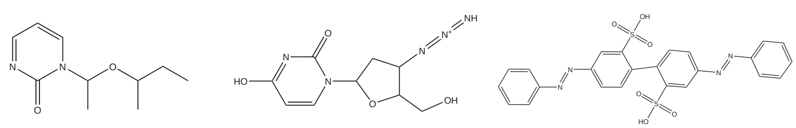
fetch index

number of candidates

REMARK: make $|C_q|$ as small as possible

Path-based Approach (Shasha, et al. PODS'02)

GRAPH DATABASE



(a) (b) (c)

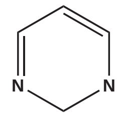
PATHS

0-length: C, O, N, S
1-length: C-C, C-O, C-N, C-S, N-N, S-O
2-length: C-C-C, C-O-C, C-N-C, ...
3-length: ...

Built an inverted index between paths and graphs

Path-based Approach (cont.)

QUERY GRAPH

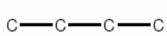


0-edge: S_C={a, b, c}, S_N={a, b, c}
1-edge: $S_{C\text{-}C}$={a, b, c}, $S_{C\text{-}N}$={a, b, c}
2-edge: $S_{C\text{-}N\text{-}C}$ = {a, b}, …
…

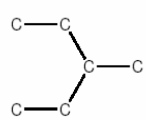
Intersect these sets, we obtain the candidate answers - graph (a) and graph (b) - which may contain this query graph.

Problems: Path-based Approach

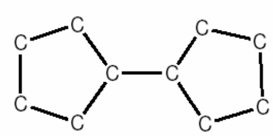
GRAPH DATABASE



(a)

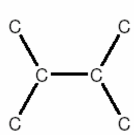


(b)



(c)

QUERY GRAPH



Only graph (c) contains this query graph. However, if we only index paths: C, C-C, C-C-C, C-C-C-C, we cannot prune graphs (a) and (b).

Using Frequent Patterns!!! (Yan et al. SIGMOD'04)

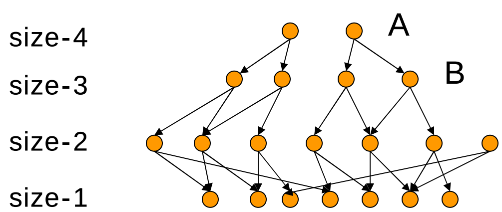
discriminative (~10^3)

frequent (~10^5)

all of the substructures (>10^7)

Discriminative Graphs

Remark: It is a kind of pattern post processing



patterns

Discriminative Graphs

- Pinpoint the most useful frequent structures
  - Given a set of structures $f_1, f_2, \ldots f_n$ and a new structure x , we measure the extra indexing power provided by x ,

$$P(x | f_1, f_2, \ldots f_n), f_i \subset x.$$

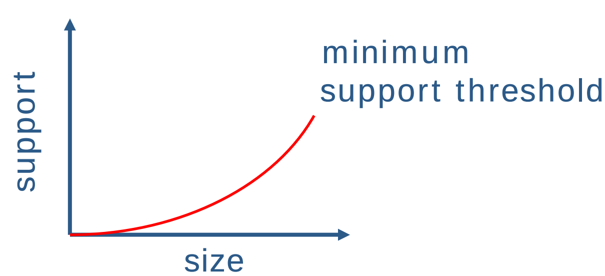
When P is small enough, x is a discriminative structure and should be included in the index

- Index discriminative frequent structures only - Reduce the index size by an order of magnitude

 Mining and Searching Graphs and Structures 73

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

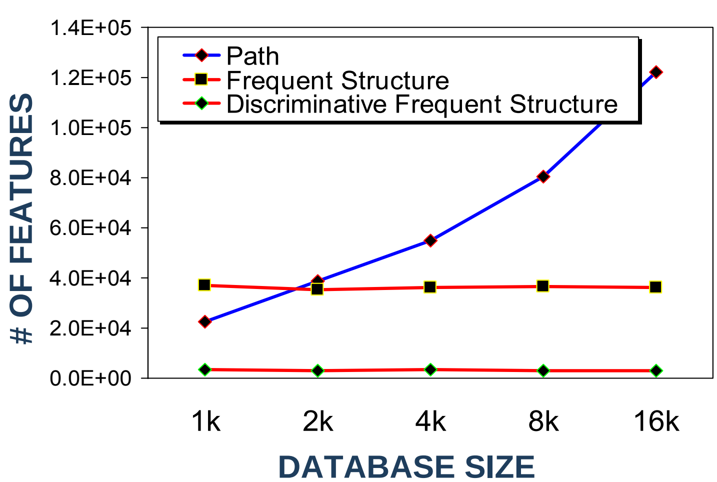


 Mining and Searching Graphs and Structures 74

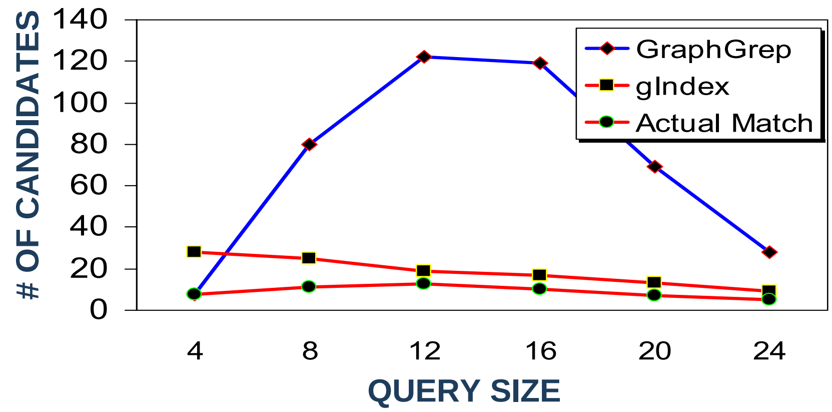
Index Graphs by Data Mining

- Identify frequent structures in the database
- Create a pattern lattice, Prune redundant frequent structures to obtain a small set of discriminative structures
- Create an inverted index between discriminative frequent structures and graphs in the database

Experiments: Index Size



Experiments: Answer Set Size

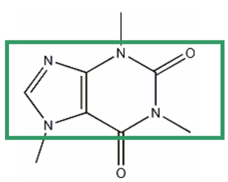


Outline

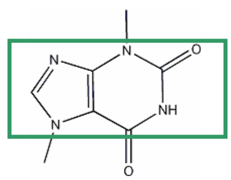
- Scalable pattern mining in graph data sets
  - Frequent subgraph pattern mining
  - Constraint-based graph pattern mining
  - Pattern summarization / selection
  - Graph clustering, classification, and compression
- Searching graph databases
  - Graph indexing methods
  - Substructure similarity search
  - Search with constraints
- Application and exploration with graph mining
  - Biological and social network analysis
  - Mining software systems: bug isolation & performance tuning
- Conclusions

Structure Similarity Search

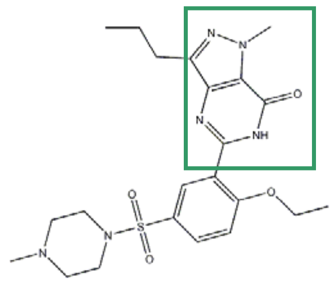
- CHEMICAL COMPOUNDS



(a) caffeine

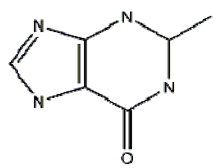


(b) diurobromine



(c) viagra

- QUERY GRAPH



Similarity Measure

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

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

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

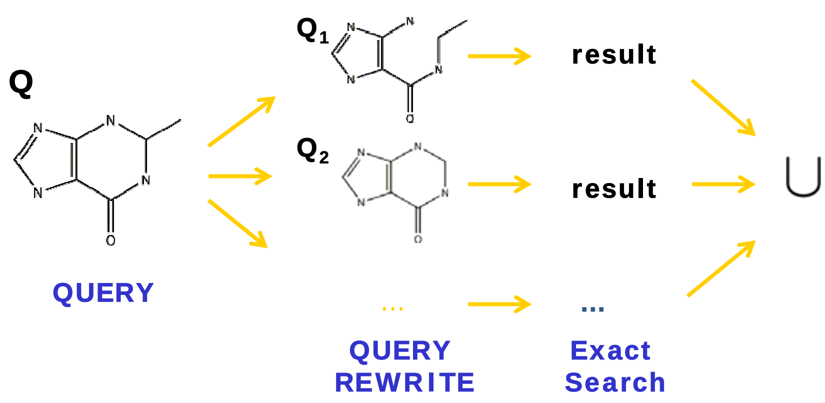
Similarity Measure

- Structure-based similarity measure
  - The maximum common subgraph (P) between query graph (Q) and target graph (G)

$$similarity = \frac{|P|}{|Q|}$$

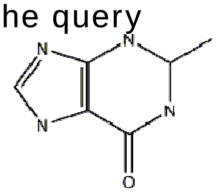
  - Similarity search: form P by deleting edges/nodes from Q; find graphs that contain P

Structure-based Similarity Measure



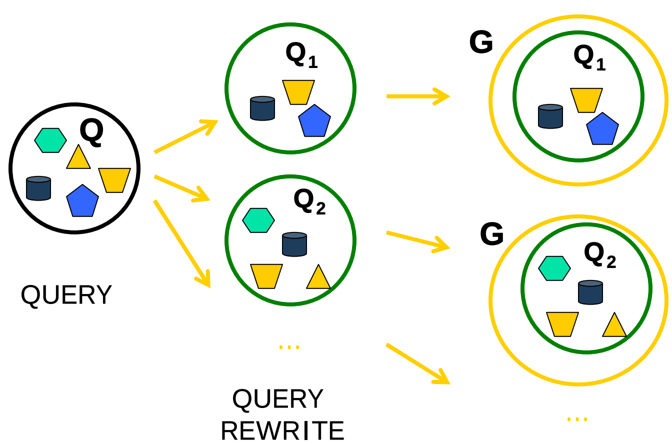
Some “Straightforward” Methods

- 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

From Edge Misses To Feature Misses



At least 3 of 5 features should be retained

Feature-based Pruning

Feature-Graph Matrix

features

	G_1	G_2	G_3	G_4	G_5
f_1	0	1	0	1	1
f_2	0	1	0	0	1
f_3	1	0	1	1	1
f_4	1	0	0	0	1
f_5	0	0	1	1	0

Assume a query graph has 5 features;
At least 3 features should be retained

Feature Miss Estimation

- Connection to maximum coverage
  - If we allow *k* edges to be relaxed (relabel or deletion), *J* is the maximum number of features to be hit by *k* edges - maximum coverage problem
- NP-complete
- A greedy algorithm exists

$$J_{greedy} \geq (1 - (1 - \frac{1}{k})^k) \cdot J$$

Feature Selection

Should we use all the features in a query graph?

- Features differentiate with selectivity and size
- How to select a good feature set?
  - features with similar properties: clustering
  - enough number of features

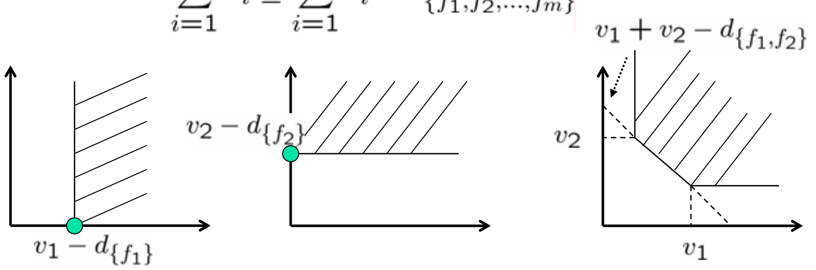
Remark: another kind of pattern post processing

Linear Inequality System

frequency of feature f_i in query graph v_i
in target graph x_i

maximum feature misses $d^k_{\{f_1,f_2,\ldots,f_m\}}$

$$\sum_{i=1}^{m} x_i \geq \sum_{i=1}^{m} v_i - d^k_{\{f_1,f_2,\ldots,f_m\}}$$



use feature f_1 **use feature f_2** **use feature f_1 & f_2**

Geometric Interpretation

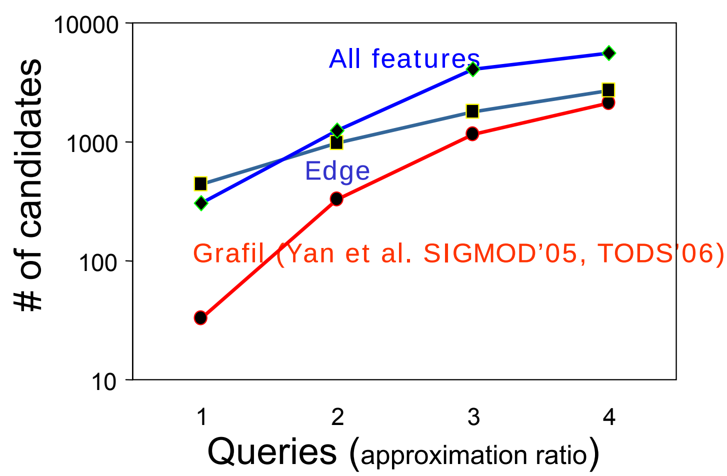
$$\sum_{i=1}^{m} x_i \geq \sum_{i=1}^{m} v_i - d^k_{\{f_1, f_2, \ldots, f_m\}}$$

$$\mathbf{Ax} \geq \mathbf{b}$$

There exist query graphs such that none of the inequalities in $Ax \geq b$ is a redundant constraint

Every halfplane defined by an inequality would cut off a polytope of nonempty volume from the convex space formed by the remaining inequalities.

Feature Selection Works

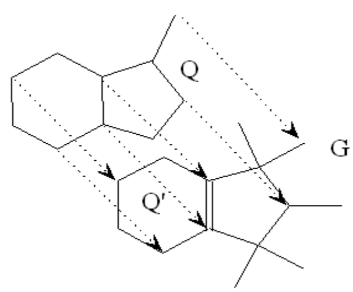


Outline

- Scalable pattern mining in graph data sets
  - Frequent subgraph pattern mining
  - Constraint-based graph pattern mining
  - Pattern summarization / selection
  - Graph clustering, classification, and compression
- Searching graph databases
  - Graph indexing methods
  - Substructure similarity search
  - Search with constraints
- Application and exploration with graph mining
  - Biological and social network analysis
  - Mining software systems: bug isolation & performance tuning
- Conclusions

Superimposed Distance

Same Topological Structure
But different Labels



$$\mathrm{MD} = \sum_{v'=f(v)} \mathrm{D}(l(v), l'(v')) + \sum_{e'=f(e)} \mathrm{D}(l(e), l'(e'))$$

Minimum Superimposed Distance

Given two graphs, Q and G, let M be the set of subgraphs in G that are isomorphic to Q. The minimum superimposed distance between Q and G is the minimum distance between Q and Q' in M.

$$d(Q,G) = \min_{Q' \in M} d(Q,Q'),$$

where $d(Q,Q')$ is a distance function of two isomorphic graphs Q and Q'.

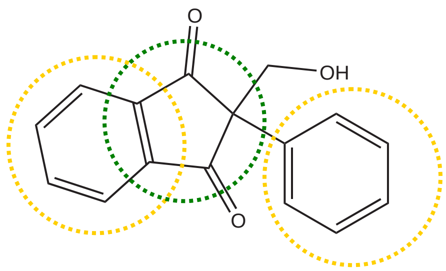
Substructure Search With Superimposed Distance

Given a set of graphs D={G_1, G_2, ..., G_n}
and a query graph Q,
SSSD is to find all G_i in D such that

$$d(Q, G_i) \leq \sigma$$

Feature-Based Index

Feature:
1. Paths (Shasha et al. PODS'02)
2. Discriminative Frequent Substructures (Yan et al. SIGMOD'04)



Partition-Based Search

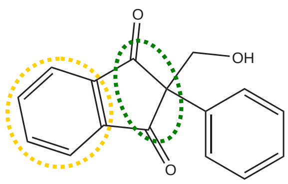
- We partition a query graph Q into non-overlapping indexed features $f_1, f_2, \ldots, f_m$, and use them to do pruning. If the distance function satisfies the following inequality,

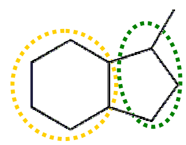
$$\sum_{i=1}^{m} d(f_i, G) \leq d(Q, G)$$

we can get the lower bound of the superimposed distance between Q and G by adding up the superimposed distance between f_i and G.

Multiple Partitions

Target graph G　　　Query graph Q

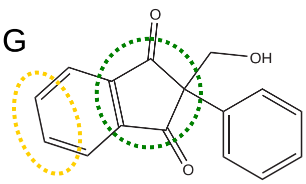




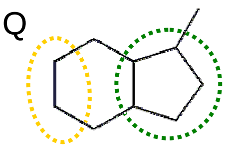
Partition I

Hexagon + Path

G



Q

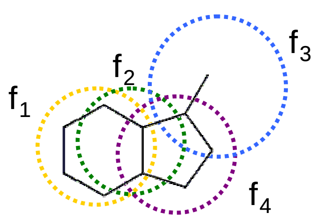


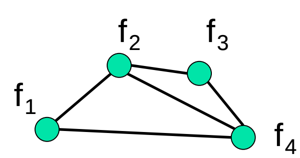
Partition II

Pentagon + Path

Overlapping Relation Graph

Query graph Q





node: feature

edge: overlapping

node weight: minimum distance between f_i and G, $d(f_i, G)$

SEARCH OPTIMIZATION

Given a graph Q=(V, E), a partition of G is a set of subgraphs $\{f_1, f_2, \ldots, f_m\}$ such that

$$V(f_i) \subseteq V \; and \; V(f_i) \cap V(f_j) = \emptyset$$

for any i!= j.

Given a graph G, optimize

$$P_{opt(Q,G)} = \arg \max_P \sum_{i=1}^{m} d(f_i, G)$$

FROM ONE TO MULTIPLE

Given a graph G, optimize

$$P_{opt(Q,G)} = \arg \max_P \sum_{i=1}^{m} d(f_i, G)$$

For one graph G, select one partition

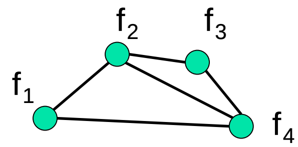
For another graph G', select another partition?

Given a set of graphs, optimize

$$P_{opt(Q,G)} = \arg \max_P \sum_{j=1}^{n} \sum_{i=1}^{m} d(f_i, G_j)$$

$$= \arg \max_P \sum_{i=1}^{m} \boxed{\sum_{j=1}^{n} d(f_i, G_j)}$$

ACROSS MULTIPLE GRAPHS



node weight is redefined

Using average minimum distance between a feature f and the graphs G_i in the database, written as

$$w(f) = \frac{\sum_{i=1}^{n} d(f, G_i)}{n}$$

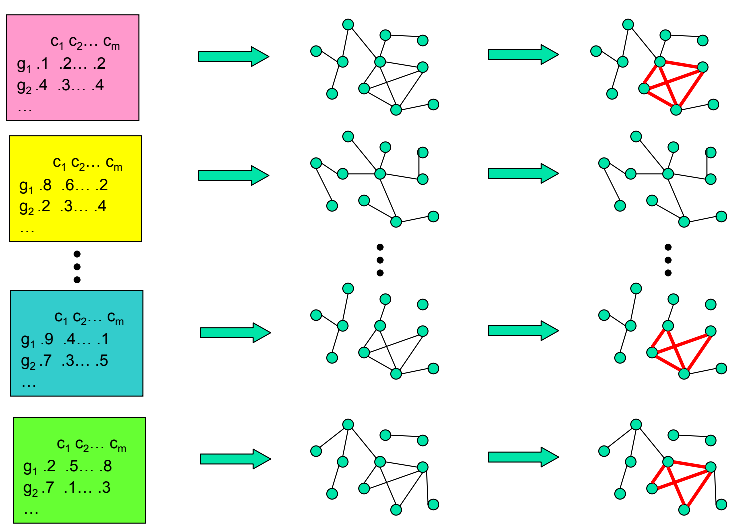
Outline

- Scalable pattern mining in graph data sets
  - Frequent subgraph pattern mining
  - Constraint-based graph pattern mining
  - Pattern summarization / selection
  - Graph clustering, classification, and compression
- Searching graph databases
  - Graph indexing methods
  - Substructure similarity search
  - Search with constraints
- Application and exploration with graph mining
  - Biological and social network analysis
  - Mining software systems: bug isolation & performance tuning
- Conclusions

Biological Networks

- Protein-protein interaction network
- Metabolic network
- Transcriptional regulatory network
- Co-expression network
- Genetic Interaction network
- ...

Mining Gene Relevance Networks



Our Solution

We develop a novel algorithm, called *CODENSE*, to mine frequent ***coherent dense*** subgraphs.

The target subgraphs have three characteristics:

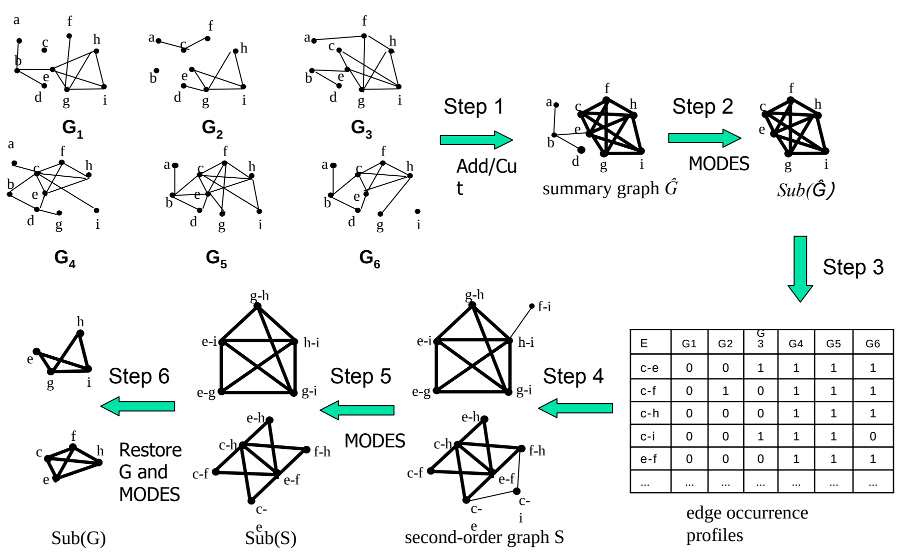
(1) All edges occur in >= k graphs (frequency)

(2) All edges should exhibit correlated occurrences in the given graph set (coherency)

(3) The subgraph is dense, where density *d* is higher than a threshold γ and $d=2m/(n(n-1))$ (density)

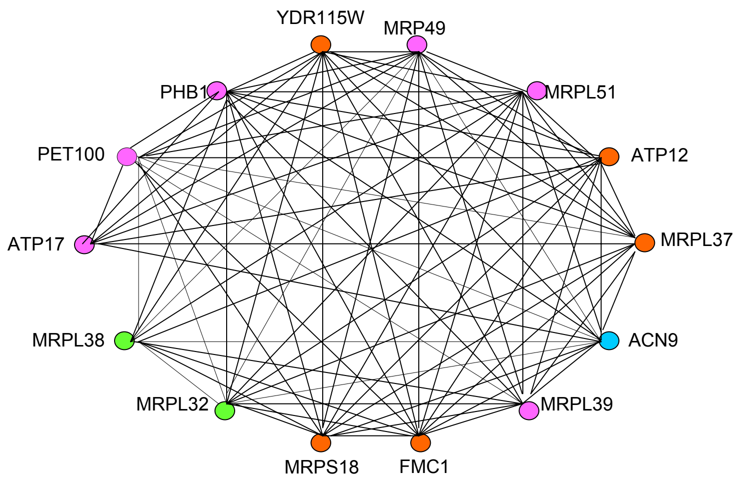
m: #edges, n: #nodes

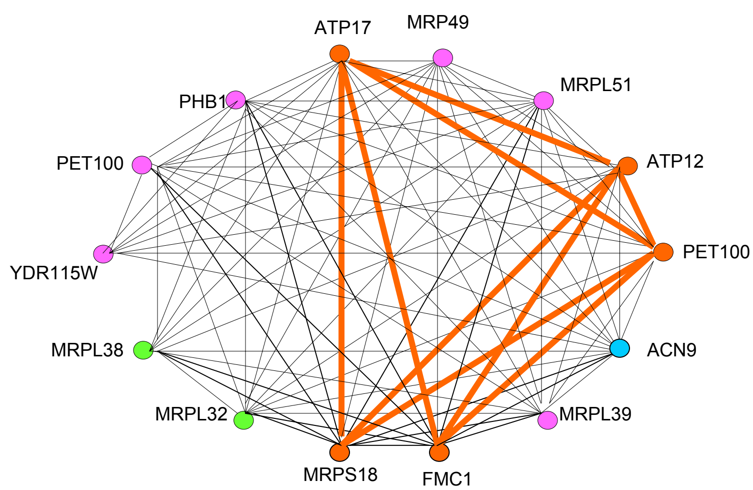
CODENSE: Mine coherent dense subgraphs



E	G1	G2	G3	G4	G5	G6
c-e	0	0	1	1	1	1
c-f	0	1	0	1	1	1
c-h	0	0	0	1	1	1
c-i	0	0	1	1	1	0
e-f	0	0	0	1	1	1
…	…	…	…	…	…	…

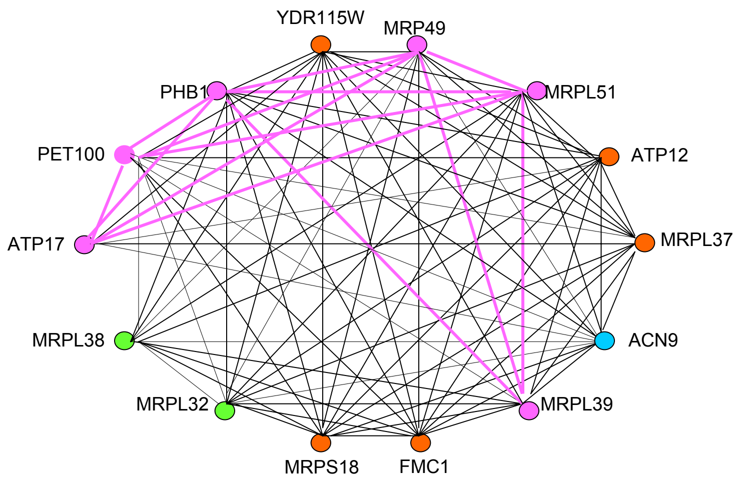
edge occurrence profiles





Yellow: YDR115W, FMC1, ATP12,MRPL37,MRPS18

GO:0019538(protein metabolism; pvalue = 0.001122)



Red:PHB1,ATP17,MRPL51,MRPL39, MRPL49, MRPL51,PET100

GO:0006091(generation of precursor metabolites and energy; pvalue=0. 001339)

Outline

- Scalable pattern mining in graph data sets
  - Frequent subgraph pattern mining
  - Constraint-based graph pattern mining
  - Pattern summarization / selection
  - Graph clustering, classification, and compression
- Searching graph databases
  - Graph indexing methods
  - Substructure similarity search
  - Search with constraints
- Application and exploration with graph mining
  - Biological and social network analysis
  - Mining software systems: bug isolation & performance tuning
- Conclusions

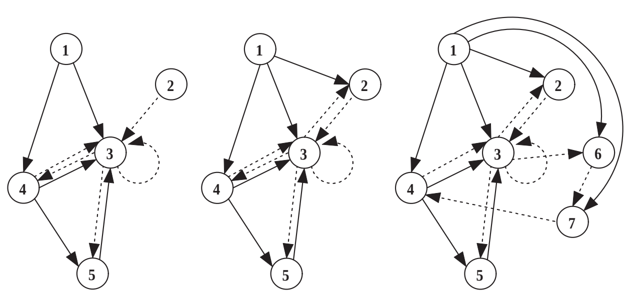
Debug Assistance Via Graph Mining

```
void subline(char *lin, char *pat, char *sub)
{
    int i, lastm, m;
    lastm = -1;
    i = 0;
    while((lin[i] != ENDSTR)) {
        m = amatch(lin, i, pat, 0);
        if ((m >= 0) && (lastm != m) ){
            putsub(lin, i, m, sub);
            lastm = m;
        }
        if ((m == -1) || (m == i)){
            fputc(lin[i], stdout);
            i = i + 1;
        } else
            i = m;
    }
}
```

- No memory violations
- No explicit errors

Program Call Graph

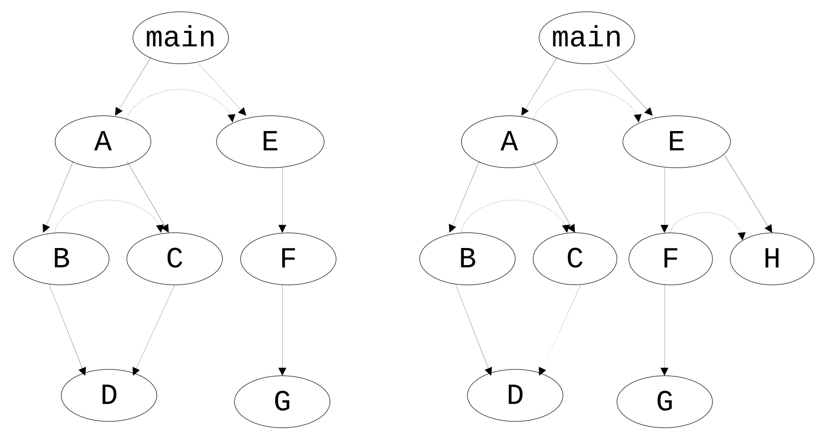
PROGRAM CALLER/CALLEE GRAPH



(1) (2) (3)

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

Program Call Graph Comparison



One Correct Execution **One Incorrect Execution**

Classification Accuracy Boost

- Check the change of classification error with or without one function in the calling graph

entrance accuracy

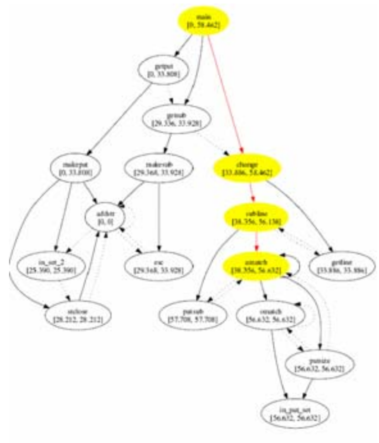
function ***F***

exit accuracy

- The difference between entrance and exit – accuracy boost

Automated Bug Isolation

```
void
subline(char *lin, char *pat, char *sub)
{
  int i, lastm, m; 8
  lastm = -1;
  i = 0;
  while ((lin[i] != ENDSTR)) {
      m= amatch(lin, i, pat, 0);
      if (m >= 0) /* && (lastm != m) BUG!!!*/{
          putsub(lin, i, m, sub);
          lastm = m;
      }
      if ((m == -1) || (m == i)){
          fputc(lin[i],stdout);
          i = i + 1;
      } else
          i = m;
  }
}
```



Replace: regular expression matching and substitution;
via http://www-static.cc.gatech.edu/aristotle/ (led by Prof. Mary Jean Harrold)

Outline

- Scalable pattern mining in graph data sets
  - Frequent subgraph pattern mining
  - Constraint-based graph pattern mining
  - Pattern summarization / selection
  - Graph clustering, classification, and compression
- Searching graph databases
  - Graph indexing methods
  - Substructure similarity search
  - Search with constraints
- Application and exploration with graph mining
  - Biological and social network analysis
  - Mining software systems: bug isolation & performance tuning
- Conclusions

Conclusions

- Graph mining has wide applications
- Frequent and closed subgraph mining methods
  - gSpan and CloseGraph: pattern-growth depth-first search approach
- Graph indexing techniques
  - Frequent and discirminative subgraphs are high-quality indexing features
- Similarity search in graph databases
  - Indexing and feature-based matching
- Biological network analysis
  - Mining coherent, dense, multiple biological networks
- Program flow analysis

Acknowledgement

Jiawei Han - UIUC
Philip S. Yu – IBM
Jasmine X. Zhou - USC
Chao Liu -UIUC
Hong Cheng - UIUC
Dong Xin - UIUC
Feida Zhu - UIUC

References (1)

- T. Asai, et al. “Efficient substructure discovery from large semi-structured data”, SDM'02
- F. Afrati, A. Gionis,and H. Mannila, “Approximating a Collection of Frequent Sets”, KDD’04
- C. Borgelt and M. R. Berthold, “Mining molecular fragments: Finding relevant substructures of molecules”, ICDM'02
- D. Cai, Z. Shao, X. He, X. Yan, and J. Han, “Community Mining from Multi-Relational Networks”, PKDD'05.
- M. Deshpande, M. Kuramochi, and G. Karypis, “Frequent Sub-structure Based Approaches for Classifying Chemical Compounds”, ICDM 2003
- M. Deshpande, M. Kuramochi, and G. Karypis. “Automated approaches for classifying structures”, BIOKDD'02
- L. Dehaspe, H. Toivonen, and R. King. “Finding frequent substructures in chemical compounds”, KDD'98
- C. Faloutsos, K. McCurley, and A. Tomkins, “Fast Discovery of 'Connection Subgraphs”, KDD'04
- H. Fröhlich, J. Wegner, F. Sieker, and A. Zell, “Optimal Assignment Kernels For Attributed Molecular Graphs”, ICML’05
- T. Gärtner, P. Flach, and S. Wrobel, “On Graph Kernels: Hardness Results and Efficient Alternatives”, COLT/Kernel’03

References (2)

- L. Holder, D. Cook, and S. Djoko. “Substructure discovery in the subdue system”, KDD'94
- J. Huan, W. Wang, D. Bandyopadhyay, J. Snoeyink, J. Prins, and A. Tropsha. “Mining spatial motifs from protein structure graphs”, RECOMB’04
- J. Huan, W. Wang, and J. Prins. “Efficient mining of frequent subgraph in the presence of isomorphism”, ICDM'03
- H. Hu, X. Yan, Yu, J. Han and X. J. Zhou, “Mining Coherent Dense Subgraphs across Massive Biological Networks for Functional Discovery”, ISMB'05
- A. Inokuchi, T. Washio, and H. Motoda. “An apriori-based algorithm for mining frequent substructures from graph data”, PKDD'00
- C. James, D. Weininger, and J. Delany. “Daylight Theory Manual Daylight Version 4.82”. Daylight Chemical Information Systems, Inc., 2003.
- G. Jeh, and J. Widom, “Mining the Space of Graph Properties”, KDD'04
- H. Kashima, K. Tsuda, and A. Inokuchi, “Marginalized Kernels Between Labeled Graphs”, ICML’03
- M. Koyuturk, A. Grama, and W. Szpankowski. “An efficient algorithm for detecting frequent subgraphs in biological networks”, Bioinformatics, 20:I200--I207, 2004.
- T. Kudo, E. Maeda, and Y. Matsumoto, “An Application of Boosting to Graph Classification”, NIPS’04

References (3)

- C. Liu, X. Yan, H. Yu, J. Han, and P. S. Yu, "Mining Behavior Graphs for 'Backtrace'' of Noncrashing Bugs''', SDM'05
- M. Kuramochi and G. Karypis. "Frequent subgraph discovery", ICDM'01
- M. Kuramochi and G. Karypis, "GREW: A Scalable Frequent Subgraph Discovery Algorithm", ICDM'04
- P. Mahé, N. Ueda, T. Akutsu, J. Perret, and J. Vert, "Extensions of Marginalized Graph Kernels", ICML'04
- B. McKay. Practical graph isomorphism. Congressus Numerantium, 30:45--87, 1981.
- S. Nijssen and J. Kok. A quickstart in frequent structure mining can make a difference. KDD'04
- J. Prins, J. Yang, J. Huan, and W. Wang. "Spin: Mining maximal frequent subgraphs from graph databases". KDD'04
- D. Shasha, J. T.-L. Wang, and R. Giugno. "Algorithmics and applications of tree and graph searching", PODS'02
- J. R. Ullmann. "An algorithm for subgraph isomorphism", J. ACM, 23:31--42, 1976.
- N. Vanetik, E. Gudes, and S. E. Shimony. "Computing frequent graph patterns from semistructured data", ICDM'02

References (4, incomplete)

- N. Wale and G. Karypis, "Acyclic Subgraph based Descriptor Spaces for Chemical Compound Retrieval and Classification", Univ. of Minnesota, Technical Report: #06–008
- C. Wang, W. Wang, J. Pei, Y. Zhu, and B. Shi. "Scalable mining of large disk-base graph databases", KDD'04
- T. Washio and H. Motoda, "State of the art of graph-based data mining", SIGKDD Explorations, 5:59-68, 2003
- X. Yan and J. Han, "gSpan: Graph-Based Substructure Pattern Mining", ICDM'02
- X. Yan and J. Han, "CloseGraph: Mining Closed Frequent Graph Patterns", KDD'03
- X. Yan, P. S. Yu, and J. Han, "Graph Indexing: A Frequent Structure-based Approach", SIGMOD'04
- X. Yan, X. J. Zhou, and J. Han, "Mining Closed Relational Graphs with Connectivity Constraints", KDD'05
- X. Yan, P. S. Yu, and J. Han, "Substructure Similarity Search in Graph Databases", SIGMOD'05
- X. Yan, F. Zhu, J. Han, and P. S. Yu, "Searching Substructures with Superimposed Distance", ICDE'06
- M. Zaki. "Efficiently mining frequent trees in a forest", KDD'02