

Data Mining: Foundation, Techniques and Applications

Lesson 13: Mining and Searching Graphs



Li Cuiping(李翠平)
School of Information
Renmin University of China



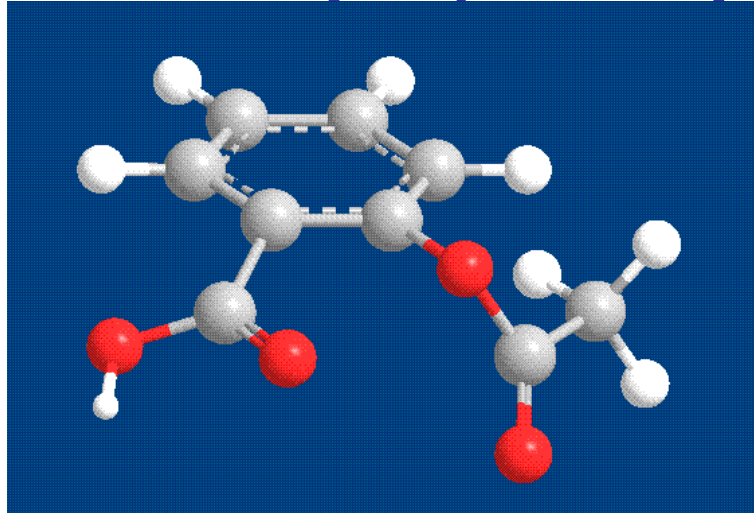
Anthony Tung(鄧錦浩)
School of Computing
National University of Singapore



Outline

- **Introduction**
- **Foundation**
 - Graph Similarity Function
 - Graph Kernels
- **Technique**
 - Graph Pattern Mining
- **Applications**
 - Graph Indexing
 - Graph Summarization for Keyword Search

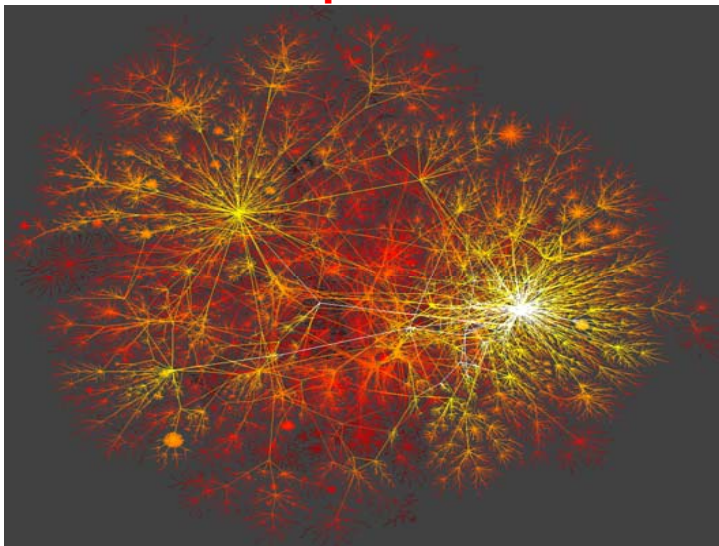
Graph, Graph, Everywhere



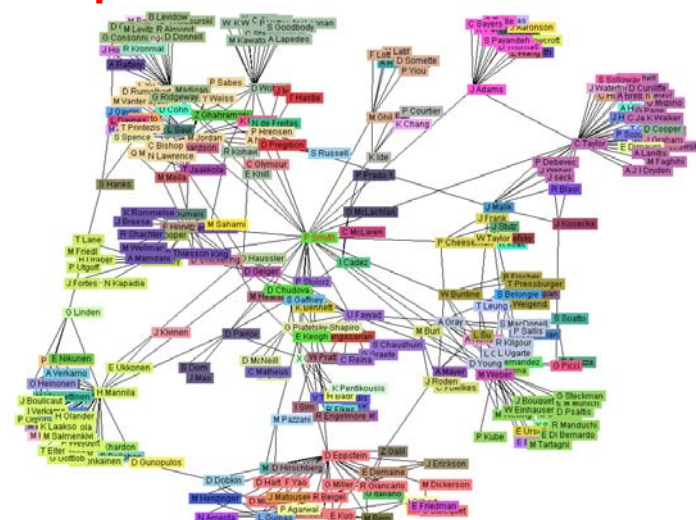
Aspirin



Yeast protein interaction network



Internet



Co-author network

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



Types of Graphs

- Single large graph vs multiple smaller graphs
- Mostly symbolic but can contain numerical values as well. Most have repeated values.
- Direct vs Undirected
- Ordered vs Unordered
- Hypergraph
 - An edge can join multiple nodes



Outline

- Introduction
- Foundation
 - Graph Similarity Function
 - Graph Kernels
- Technique
 - Graph Pattern Mining
- Applications
 - Graph Indexing
 - Graph Summarization for Keyword Search



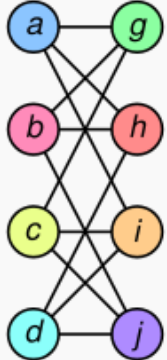
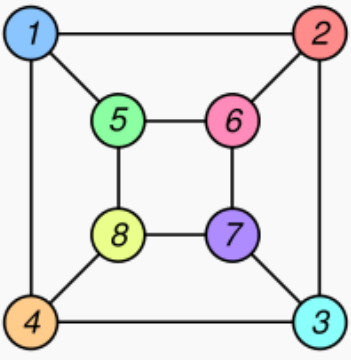
Maximum Common Subgraph

- Problems:
 - To compare objects represented as graphs and to determine the degree and composition of the similarity between the objects
- Applications:
 - Chemistry, biology, computer vision and image recognition, etc.

Definition & Terminology

■ Isomorphic

- Two graphs are said to be isomorphic if there is a one-to-one correspondence between their vertices and an edge exists between two vertices in one graph if an edge exists between the two corresponding vertices in the other graph.

Graph G	Graph H	An isomorphism between G and H
		$f(a) = 1$ $f(b) = 6$ $f(c) = 8$ $f(d) = 3$ $f(g) = 5$ $f(h) = 2$ $f(i) = 4$ $f(j) = 7$



Definition & Terminology

- Induced Subgraph

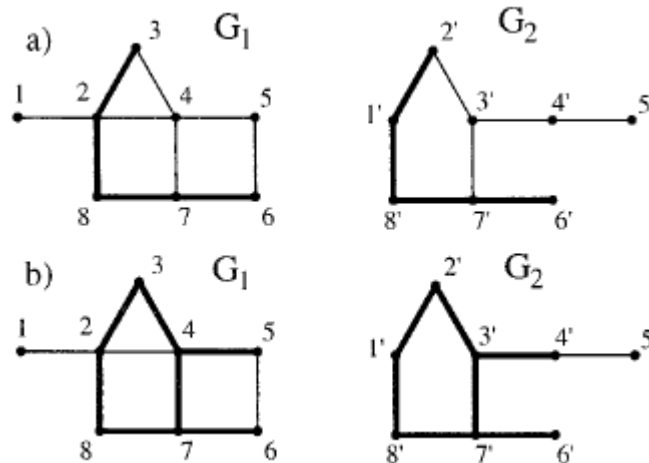
- A set S of vertices of a graph G and those edges of G with both endpoints in S .

- Common Induced Subgraph

- A graph G_{12} is a common induced subgraph of graph G_1 and G_2 if G_{12} is isomorphic to induced subgraphs of G_1 and G_2

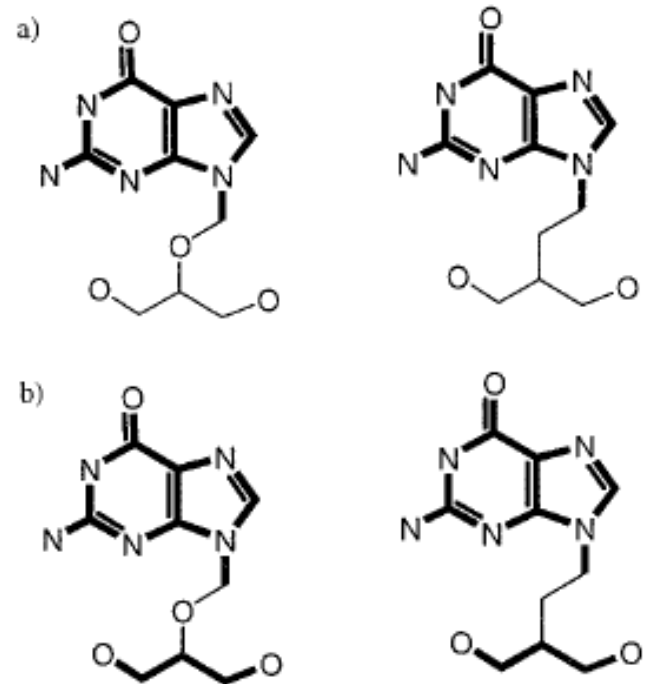
Definitions & Terminology

- Maximum Common Induced Subgraph (MCIS)
 - Subgraph consists of a graph G_{12} with the largest number of vertices meeting those property
- Maximum Common Edge Subgraph (MCES)
 - Subgraph consists of the largest number of edges common to both G_1 and G_2



Definitions & Terminology

- Connected MCS
 - Each vertex is connected to every other vertex by at least one path in the graph
- Disconnected MCS
 - Comprised of two or more subgraph components



MCES to MCIS

- An edge isomorphism between two graphs, G_1 and G_2 induces a vertex isomorphism provided that a ΔY exchange does not occur.

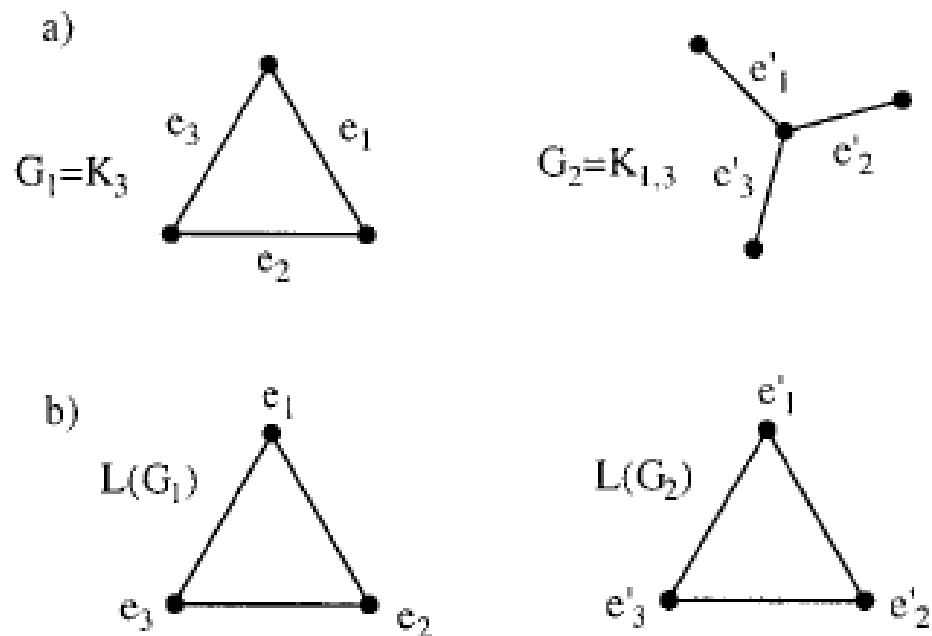


Figure 3. ΔY exchange

Algorithms

- For comparison between a pair of chemical graphs consisting of m and n atoms respectively, the maximum number of possible atom-by-atom comparison necessary to determine all common subgraphs consisting of k atoms is

$$\frac{m!n!}{(m-k)!(n-k)!k!}$$

Classification

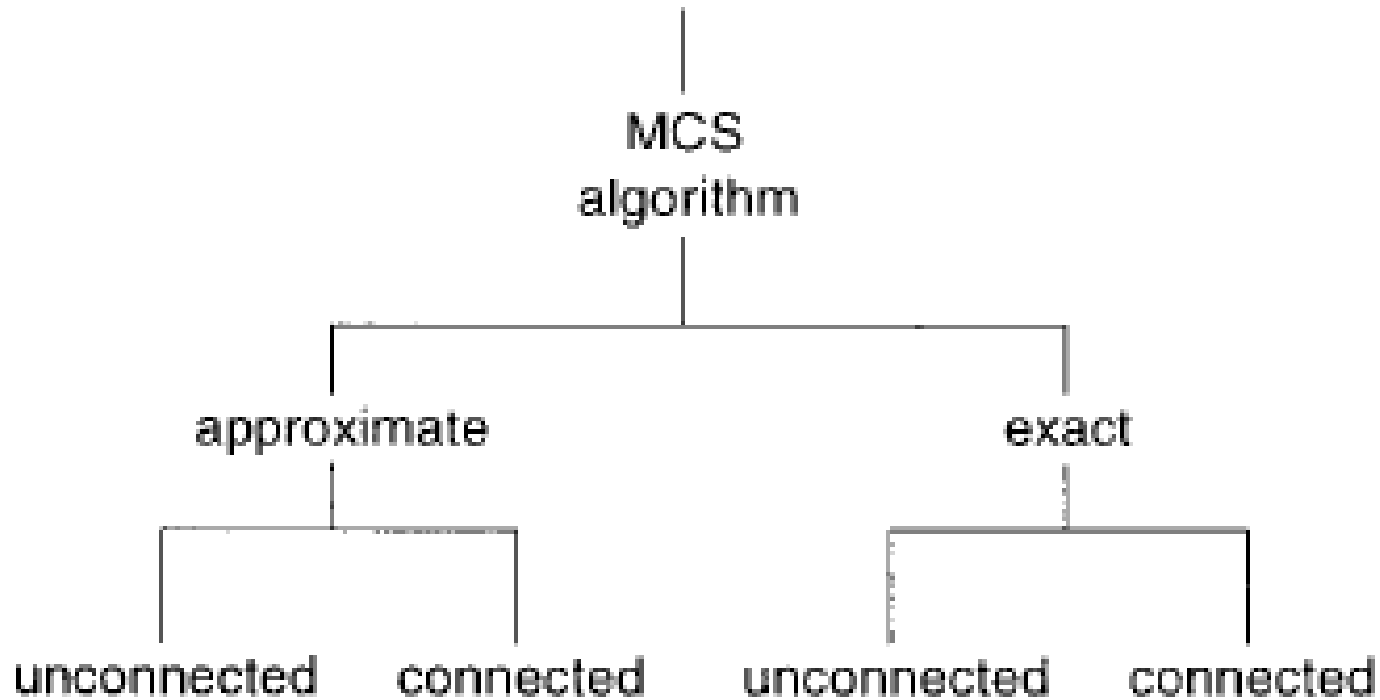


Figure 4. MCS algorithm classification



Algorithms

- Exact Algorithm
 - Maximum clique-based algorithm
 - Backtracking Algorithms
 - Dynamic Programming
- Approximate Algorithm
 - Genetic Algorithm
 - Combinatorial Optimization
 - Fragment Storage
 - Adhoc Procedures
 - 3D specific Algorithm



Maximum clique-based algorithm

- Clique – subset of vertices in the graph such that each pair of vertices in the subset is connected by an edge in the graph G
- Maximum Clique – largest such subset present in the graph
- MCIS problem is reduced to maximum clique problem by constructing a compatibility graph using adjacency properties of the graphs being compared (ie, the MCIS factor graphs)



Maximum clique-based algorithm

- The MCIS between graphs being compared is equivalent to a maximum clique in the compatibility graph
- It is also the modular product graph in mathematics

Modular Product

- Modular product of two graphs G_1 and G_2 is defined on the vertex set $V(G_1) \times V(G_2)$ with two vertices (u_i, v_i) and (u_j, v_j) being adjacent whenever

$$(u_i u_j) \in e(G_1) \text{ and } (v_i, v_j) \in E(G_2), \text{ or}$$

$$(u_i u_j) \notin e(G_1) \text{ and } (v_i, v_j) \notin E(G_2).$$

u_i and u_j are adjacent in G_1 and v_i and v_j are adjacent in G_2

OR

u_i and u_j are not adjacent in G_1 and v_i and v_j are not adjacent in G_2

Modular Product

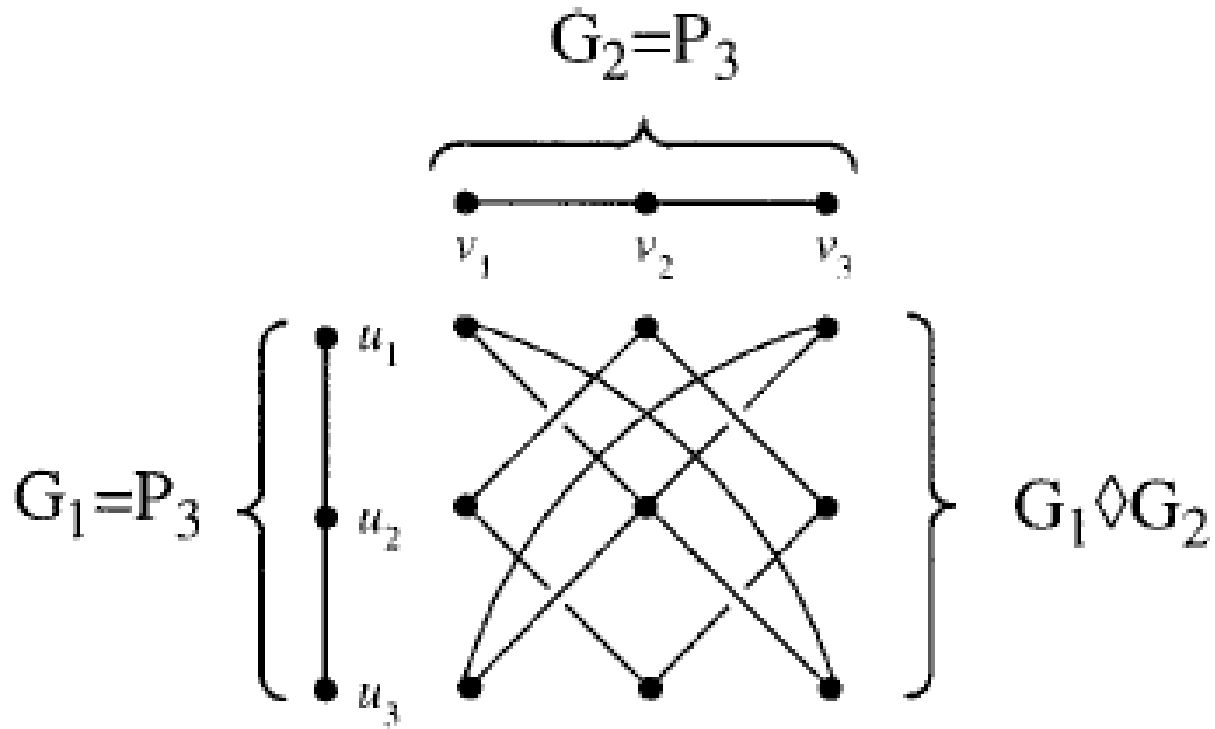


Figure 5. Modular product $P_3 \diamond P_3$



Backtracking Algorithm

- By McGregor and Wong
- Attempt to reduce the number of backtrack instances necessary by inspecting the set of possible solutions remaining at some point in the depth-first search.
- Then determine whether it is necessary to extend the current solution
- The set of possible solutions is evaluated by enforcing a connectivity relation with the currently detected solution



Dynamic Programming

- Calculate the connected MCES from a set of factor graphs.
- Almost tree of bounded degree is a graph G such that $|E(B)| \leq |V(B)| + K$ holds for every biconnected component B of G , where K is a constant.
- Biconnected component – maximal edge induced subgraph in a connected graph such that the subgraph cannot be disconnected by eliminating a vertex

Dynamic Programming

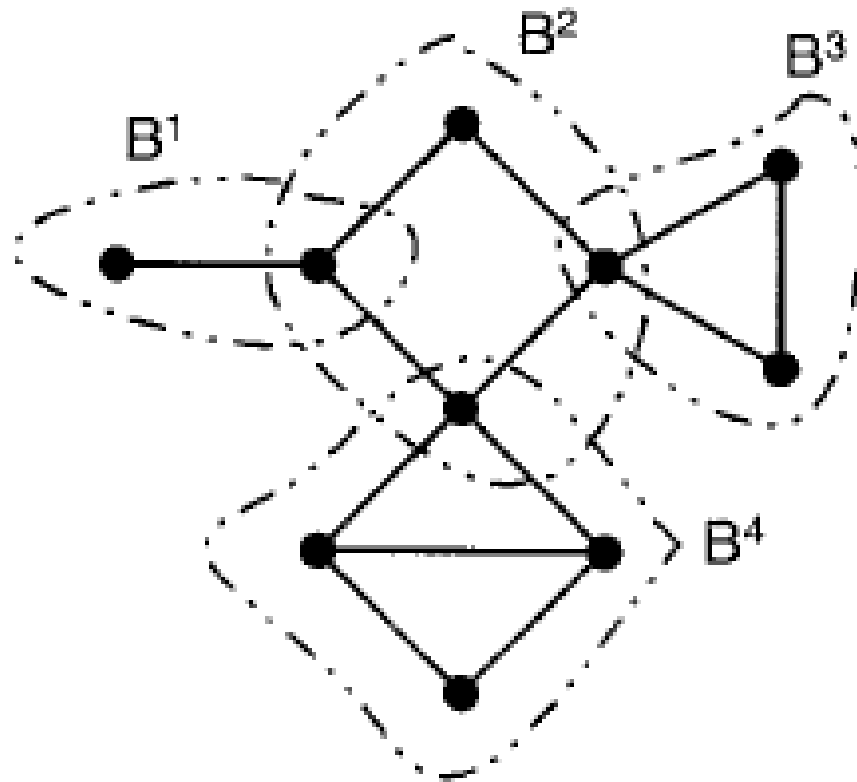


Figure 6. Biconnected components



Algorithms

- Exact Algorithm
 - Maximum clique-based algorithm
 - Backtracking Algorithms
 - Dynamic Programming
- Approximate Algorithm
 - Genetic Algorithm
 - Combinatorial Optimization
 - Fragment Storage
 - Adhoc Procedures
 - 3D specific Algorithm

Genetic Algorithm

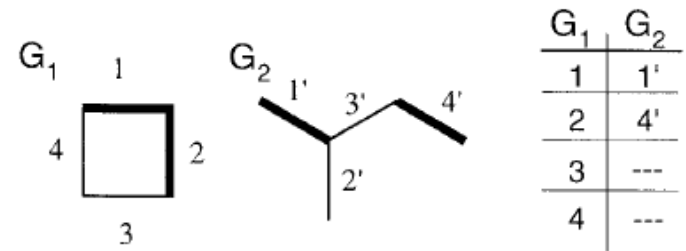
- Survival of the fittest based on a fitness function
- For chemical graphs,

$$F = N - V - (T_1 + T_2 - 2)$$

N = total number of bonds in the two structures that participate in a bond matching

V = how many bonds are involved in the situation where two adjacent bonds in the other structure.

T_1, T_2 = the number of unconnected subgraphs in the two graphs, respectively





Combinatorial Optimization

- By Funabiki and Kitamichi
- 2DOM (2-stage Discrete Optimization Method)
- Stage 1: Problem Construction Stage
 - Greedy matching between the graphs being compared
- Stage 2: Refinement stage
 - Uses a randomized, discrete descent method to minimize an objective function consisting of the number of unmatched edges in the factor graph with the fewest edges



Fragment Storage

- Feasible only with database searching
- The database to be searched is stored in a multi-level tree where each bifurcation point in the tree corresponds to particular chemical substructure.
- Increasing a level in the tree to a lower bifurcation point, corresponds to adding a specific chemical substructure fragment to the substructure represented by the preceding bifurcation point.



Fragment Storage

- Using the multi-leveled database structure, it is then possible to perform rapid similarity searching of pre-processed databases.
- Using the query compound as a template, the fragment tree is traversed until a bifurcation point is reached where it is not possible to continue.
- The substructure represented by this bifurcation point corresponds to an approximation of the MCES between the query compound and all of the database compounds located lower in the search tree.



Ad hoc Procedures

- Graph Walking procedure
 - Determine connected substructure
 - Involves 'growing' a currently detected subgraph by adding a vertex and all edges incident between the current subgraph and the newly selected vertex

Problem Reduction

- A graph is an abstract concept and the vertices and edges do not necessarily have to correlate directly with atoms, bonds, and distance ranges in chemical structure
- Simplified graph to represent nodes and edges

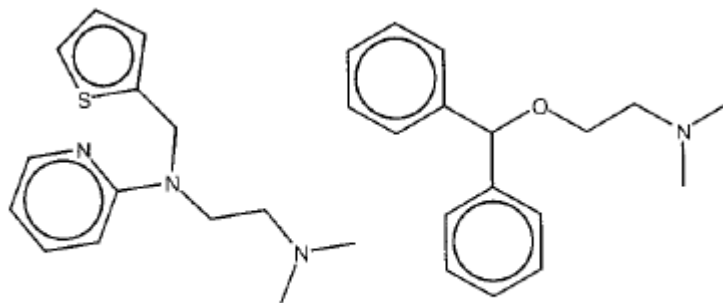


Figure 8. Chemically similar?



Problem Reduction

- Advantages:-
 - 1. If the modified graph contains fewer nodes and edges, it may significantly increase the efficiency of MCS comparison
 - 2. It may reflect a more desirable understanding of the similarity between the structures being compared



Problem Reduction

- Examples:
- Superatoms
 - Vertices can be one of a predefined number of ring system, functional groups or alkyl chains
 - An edge exists if pair of superatoms are adjacent in the molecular structure.
- Feature trees
 - Chemical graph is reduced to tree graph (graph without rings or cycles)
 - Use split search and match search



Screening Procedures

- If a user specifies a lower-bound for a particular similarity comparison, then an effective upper-bound estimate based on the MCS concept can provide a means of screening comparisons that cannot potentially result in an MCS exceeding the specified lower bound.

Screening Procedures

- Asymmetric similarity co-efficient, $S_{lb} = N_c / N_q$
 - N_c = number of bond pairs in common between a query and database graph
 - N_q = number of edges in the query structure
- If it is found prior to graph matching that $N_c < S_{lb} \cdot N_q$ after specifying a minimum acceptable value of S_{lb} , then it is not necessary to proceed to a rigorous graph matching

Screening Procedures

- Two levels of screening
- Similarity coefficient,

$$S_{ub} = \frac{|G_{12}|^2}{|G_1| \times |G_2|},$$

$|G_1|$ and $|G_2|$ = number of vertices and edges in graph G_1 and G_2

G_{12} = upper bound estimate given for the MCS between G_1 and G_2



Outline

- Introduction
- Foundation
 - Graph Similarity Function
 - Graph Kernels
- Technique
 - Graph Pattern Mining
- Applications
 - Graph Indexing
 - Graph Summarization for Keyword Search



Kernel Function

- Inner product of two feature vectors
- This paper defines a kernel function between graphs
- Previous methods:
 - Decompose graph into substructures
 - Feature vector is composed of the counts of these substructures
 - Limit dimensionality of vectors by setting threshold on substructure length or frequency
 - Use methods from dynamic programming



Kernel function

In this paper:

- Is defined on infinite dimensional path count vectors
- Each label path is produced by random walks on graph
- Is defined as the inner product of the count vectors averaged over all possible label paths

Marginalized Kernels

Marginalized Kernels

$$K(\mathbf{x}, \mathbf{x}') = \sum_{\mathbf{h}} \sum_{\mathbf{h}'} K_z(\mathbf{z}, \mathbf{z}') p(\mathbf{h}|\mathbf{x}) p(\mathbf{h}'|\mathbf{x}').$$

- \mathbf{h} and \mathbf{h}' are the hidden variables (sequence of vertex indices) obtained by random walks on graph \mathbf{x} and \mathbf{x}' respectively
- K_z is a joint kernel between the sequences of vertex and edge labels traversed in the random walk
- $K(\mathbf{x}, \mathbf{x}')$ is the *marginalized kernel* between graph \mathbf{x} and \mathbf{x}' , defined as the expectation of K_z over all possible values of \mathbf{h} and \mathbf{h}'

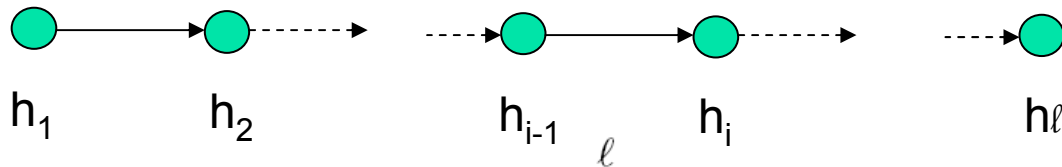


Definition

- $|G|$ = the number of vertices in graph G
- Let $v_i \in \Sigma_v$ denote the label of vertex i
- Let $e_{ij} \in \Sigma_E$ denote the label of the edges from i to j
- The task is to define $K(G, G')$ between two labeled graphs G and G'

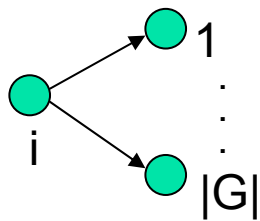
Random Walks on Graph, $p(h|G)$

- $h = (h_1, \dots, h_\ell)$ is a sequence of vertex indices obtained by a random walk on graph G



$$p(\mathbf{h}|G) = p_s(h_1) \prod_{i=2}^{\ell} p_t(h_i|h_{i-1}) p_q(h_\ell)$$

- $p_q(i)$ is the probability that the random walk ends at vertex i



$$\sum_{j=1}^{|G|} p_t(j|i) + p_q(i) = 1.$$

Joint Kernel, K_z

- When random walk is done as described in h , the traversed labels are:

$$v_{h_1} e_{h_1 h_2} v_{h_2} e_{h_2 h_3} v_{h_3} \dots$$

- Joint kernel between two label paths z and z'

$$z = v_{h_1} e_{h_1 h_2} v_{h_2} e_{h_2 h_3} v_{h_3} \dots v_{h_\ell}$$

$$z' = v'_{h'_1} e'_{h'_1 h'_2} v'_{h'_2} e'_{h'_2 h'_3} v'_{h'_3} \dots v'_{h'_\ell}$$

$$K_z(z, z') = \begin{cases} 0 & (\ell \neq \ell') \\ K(v_{h_1}, v'_{h'_1}) \prod_{i=2}^{\ell} K(e_{h_{i-1} h_i}, e'_{h'_{i-1} h'_i}) \times K(v_{h_\ell}, v'_{h'_\ell}) & (\ell = \ell') \end{cases}$$

Marginalized Kernel, $K(G, G')$

- The expectation of K_z over all possible (values and length of) \mathbf{h} and \mathbf{h}' :

$$K(G, G') = \sum_{\ell=1}^{\infty} \sum_{\mathbf{h}} \sum_{\mathbf{h}'} K_z(\mathbf{z}, \mathbf{z}') p(\mathbf{h}|G) p(\mathbf{h}'|G')$$

- Plug in:

$$p(\mathbf{h}|G) = p_s(h_1) \prod_{i=2}^{\ell} p_t(h_i|h_{i-1}) p_q(h_{\ell})$$

$$K_z(\mathbf{z}, \mathbf{z}') = \begin{cases} 0 & (\ell \neq \ell') \\ K(v_{h_1}, v'_{h'_1}) \prod_{i=2}^{\ell} K(e_{h_{i-1}h_i}, e'_{h'_{i-1}h'_i}) \times \\ & K(v_{h_{\ell}}, v'_{h'_{\ell}}) & (\ell = \ell') \end{cases}$$

Marginalized Kernel, $K(G, G')$

$$\begin{aligned}
 & K(G, G') \\
 &= \sum_{\ell=1}^{\infty} \sum_{\mathbf{h}} \sum_{\mathbf{h}'} p_s(h_1) \prod_{i=2}^{\ell} p_t(h_i | h_{i-1}) p_q(h_{\ell}) \times \\
 &\quad p'_s(h'_1) \prod_{j=2}^{\ell} p'_t(h'_j | h'_{j-1}) p'_q(h'_{\ell}) \times \\
 &\quad K(v_{h_1}, v_{h'_1}) \prod_{k=2}^{\ell} K(e_{h_{k-1}h_k}, e'_{h'_{k-1}, h'_k}) K(v_{h_k}, v_{h'_k}),
 \end{aligned}$$

Where

$$\sum_{\mathbf{h}} := \sum_{h_1=1}^{|G|} \cdots \sum_{h_{\ell}=1}^{|G|}$$

Marginalized Kernel, $K(G, G')$

$$K(G, G') = \lim_{L \rightarrow \infty} \sum_{\ell=1}^L \sum_{h_1, h'_1} \boxed{s(h_1, h'_1)} \times \quad (5)$$

$$\left(\sum_{h_2, h'_2} \boxed{t(h_2, h'_2, h_1, h'_1)} \left(\sum_{h_3, h'_3} t(h_3, h'_3, h_2, h'_2) \times \right. \right. \\ \left. \left. \left(\dots \left(\sum_{h_\ell, h'_\ell} t(h_\ell, h'_\ell, h_{\ell-1}, h'_{\ell-1}) \boxed{q(h_\ell, h'_\ell)} \right) \right) \dots \right) \right)$$

$$q(h_\ell, h'_\ell) := p_q(h_\ell) p'_q(h'_\ell) p'_t(h'_\ell | h'_{\ell-1}) \times \\ r_\ell(h_1, h'_1) \cdot (v_{h_\ell}, v_{h'_\ell}) \mathbf{\Lambda} (e_{h_{\ell-1} h_\ell}, e_{h'_{\ell-1} h'_\ell})$$

Marginalized Kernel, $K(G, G')$

$$K(G, G') = \lim_{L \rightarrow \infty} \sum_{\ell=1}^L \sum_{h_1, h'_1} s(h_1, h'_1) r_\ell(h_1, h'_1),$$

Where

$$r_1(h_1, h'_1) := q(h_1, h'_1)$$

$$\begin{aligned} & r_\ell(h_1, h'_1) \\ & := \left(\sum_{h_2, h'_2} t(h_2, h'_2, h_1, h'_1) \left(\sum_{h_3, h'_3} t(h_3, h'_3, h_2, h'_2) \times \right. \right. \\ & \quad \left. \left. \left(\dots \left(\sum_{h_\ell, h'_\ell} t(h_\ell, h'_\ell, h_{\ell-1}, h'_{\ell-1}) q(h_\ell, h'_\ell) \right) \right) \dots \right) \right) \end{aligned}$$

Recursive Relationships

$$\begin{aligned} r_\ell(h_1, h'_1) &:= \left(\sum_{h_2, h'_2} t(h_2, h'_2, h_1, h'_1) \left(\sum_{h_3, h'_3} t(h_3, h'_3, h_2, h'_2) \times \right. \right. \\ &\quad \left. \left. \left(\dots \left(\sum_{h_\ell, h'_\ell} t(h_\ell, h'_\ell, h_{\ell-1}, h'_{\ell-1}) q(h_\ell, h'_\ell) \right) \dots \right) \right) \right) \\ &: \sum_{i, j} t(i, j, h_1, h'_1) r_{\ell-1}(i, j) \end{aligned}$$

Marginalized Kernel, $K(G, G')$

$$\begin{aligned} K(G, G') &= \lim_{L \rightarrow \infty} \sum_{\ell=1}^L \sum_{h_1, h'_1} s(h_1, h'_1) r_\ell(h_1, h'_1), \\ &= \sum_{h_1, h'_1} s(h_1, h'_1) \lim_{L \rightarrow \infty} \sum_{\ell=1}^L r_\ell(h_1, h'_1) \\ &= \sum_{h_1, h'_1} s(h_1, h'_1) \lim_{L \rightarrow \infty} R_L(h_1, h'_1), \end{aligned}$$

Marginalized Kernel Equilibrium

$$\begin{aligned} R_L(h_1, h'_1) &:= \sum_{\ell=1}^L r_\ell(h_1, h'_1) \\ &= r_1(h_1, h'_1) + \sum_{\ell=2}^L r_\ell(h_1, h'_1) \\ &= r_1(h_1, h'_1) + \sum_{\ell=2}^L \sum_{i,j} t(i, j, h_1, h'_1) r_{\ell-1}(i, j) \end{aligned}$$

$$= r_1(h_1, h'_1) + \sum_{i,j} t(i, j, h_1, h'_1) R_{L-1}(i, j)$$

$$R_\infty(h_1, h'_1) = r_1(h_1, h'_1) + \sum_{i,j} t(i, j, h_1, h'_1) R_\infty(i, j)$$

Marginalized Kernel

Linear Simultaneous Equation

Solve:

$$R_{\infty}(h_1, h'_1) = r_1(h_1, h'_1) + \sum_{i,j} t(i, j, h_1, h'_1) R_{\infty}(i, j)$$

Substitute to:

$$K(G, G') = \sum_{h_1, h'_1} s(h_1, h'_1) \lim_{L \rightarrow \infty} R_L(h_1, h'_1)$$

$|G||G'| \times |G||G'|$ coefficient matrix

Sparse Graph

Iterate till convergence

Marginalized Kernel

If the termination probabilities are constant (γ) over all vertices,

$\lim_{L \rightarrow \infty} R_L(h_1, h'_1)$ converges if

$$K(v, v')K(e, e') < \frac{1}{(1 - \gamma)^2}.$$

Experimental Results

■ Compare with Pattern Discovery Algorithm

Table 2. Classification accuracies (%) of the pattern discovery method. 'MinSup' shows the ratio of the minimum support parameter to the number of compounds m/n .

MinSup	MM	FM	MR	FR	MUTAG
0.5%	60.1	57.6	61.3	66.7	88.3
1 %	61.0	61.0	62.8	63.2	87.8
3 %	58.3	55.9	60.2	63.2	89.9
5 %	60.7	55.6	57.3	63.0	86.2
10 %	58.9	58.7	57.8	60.1	84.6
20%	61.0	55.3	56.1	61.3	83.5

Table 3. Classification accuracies (%) of our graph kernel. The parameter γ is the termination probability of random walks, which controls the effect of the length of label paths.

γ	MM	FM	MR	FR	MUTAG
0.1	62.2	59.3	57.0	62.1	84.3
0.2	62.2	61.0	57.0	62.4	83.5
0.3	64.0	61.3	56.7	62.1	85.1
0.4	64.3	61.9	56.1	63.0	85.1
0.5	64.0	61.3	56.1	64.4	83.5
0.6	62.8	61.9	54.4	65.8	83.0
0.7	63.1	62.5	54.1	63.2	81.9
0.8	63.4	63.4	54.9	64.1	79.8
0.9	62.8	61.6	58.4	66.1	78.7



Summary

- Using random walks to compute marginalized kernel
- Takes into account all possible label paths without computing feature values explicitly
- Can be used for sequences, trees, DAGs
- Implicit parameter is required (the termination probability of random walks)



Outline

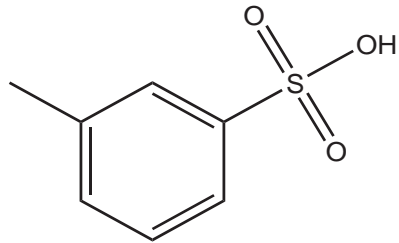
- Introduction
- Foundation
 - Graph Similarity Function
 - Graph Kernels
- Technique
 - **Graph Pattern Mining**
- Applications
 - Graph Indexing
 - Graph Summarization for Keyword Search

Graph Pattern Mining

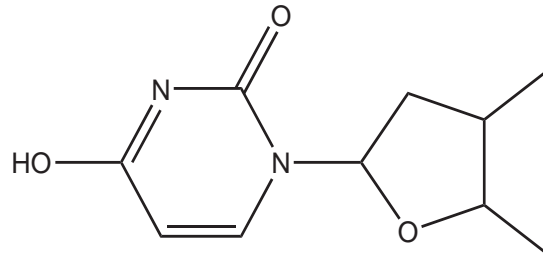
- *Frequent* subgraphs
 - A (sub)graph is *frequent* if its *support* (occurrence frequency) in a given dataset is no less than a *minimum support* threshold
- Applications of graph pattern mining
 - Mining biochemical structures
 - Program control flow analysis
 - Mining XML structures or Web communities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

Example: Frequent Subgraphs

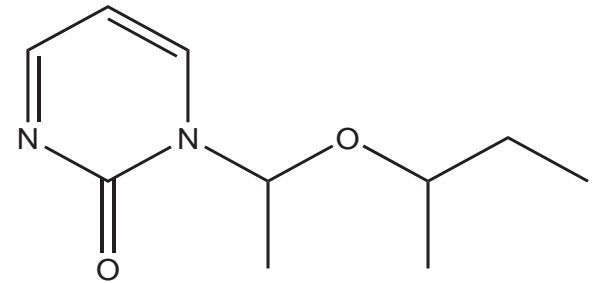
GRAPH DATASET



(A)



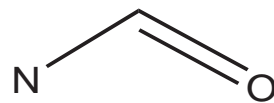
(B)



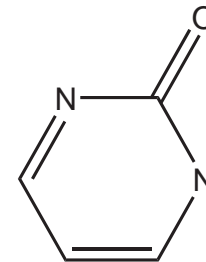
(C)

FREQUENT PATTERNS (MIN SUPPORT IS 2)

(1)

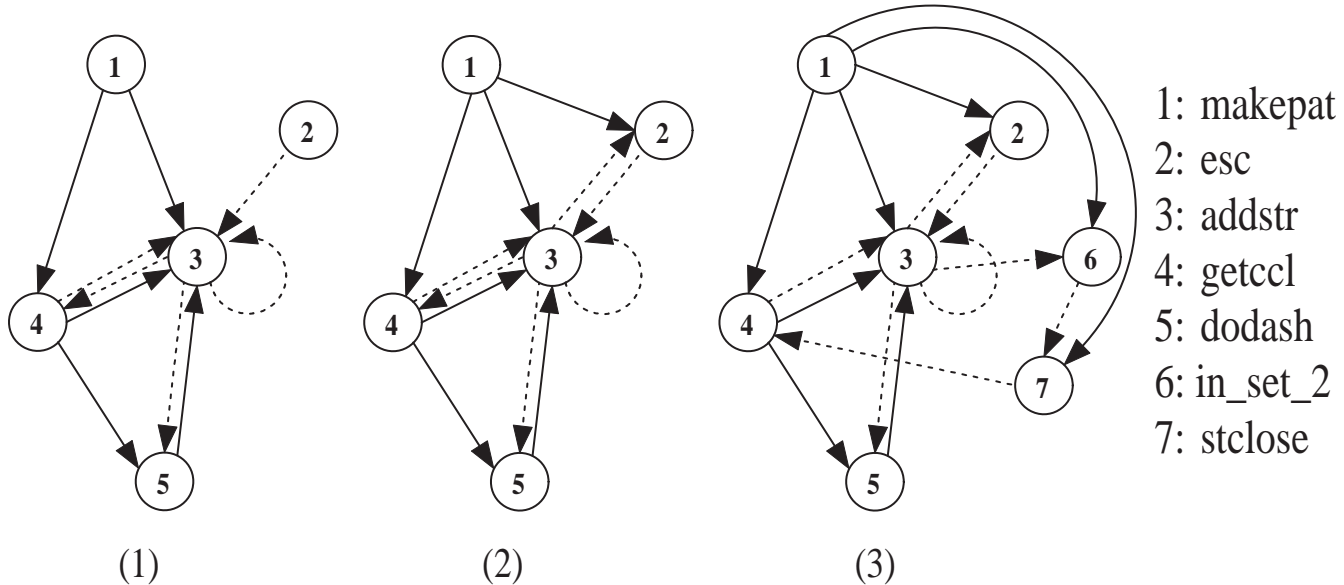


(2)

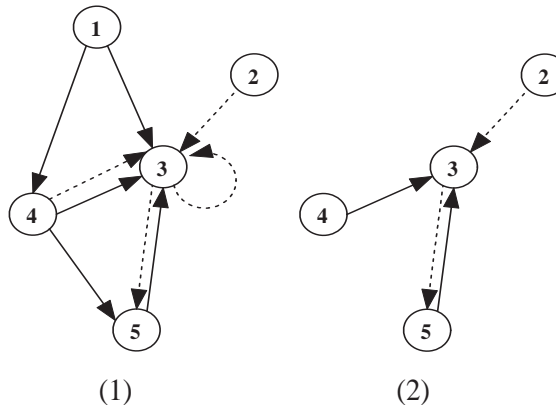


EXAMPLE (II)

GRAPH DATASET



FREQUENT PATTERNS (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

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:
 $DL(S) + DL(G \setminus S)$
- 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

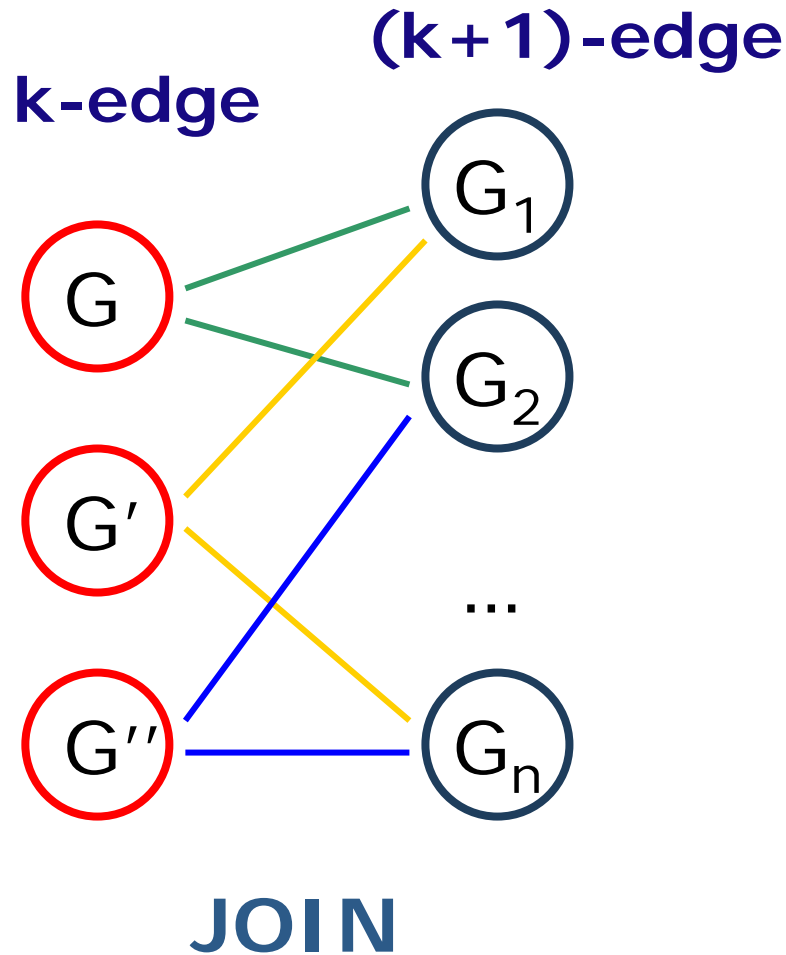
- Apriori-based approach
 - AGM/AcGM: Inokuchi, et al. (PKDD'00)
 - FSG: Kuramochi and Karypis (ICDM'01)
 - PATH[#]: Vanetik and Gudes (ICDM'02, ICDM'04)
 - FFSM: Huan, et al. (ICDM'03)
- Pattern growth approach
 - MoFa, Borgelt and Berthold (ICDM'02)
 - gSpan: Yan and Han (ICDM'02)
 - Gaston: Nijssen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



Properties of Graph Mining Algorithms

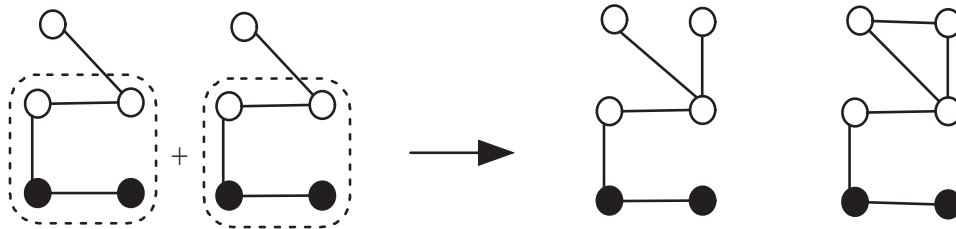
- Search order
 - breadth vs. depth
- Generation of candidate subgraphs
 - apriori vs. pattern growth
- Elimination of duplicate subgraphs
 - passive vs. active
- Support calculation
 - embedding store or not
- Discover order of patterns
 - path → tree → graph

Apriori-Based Approach



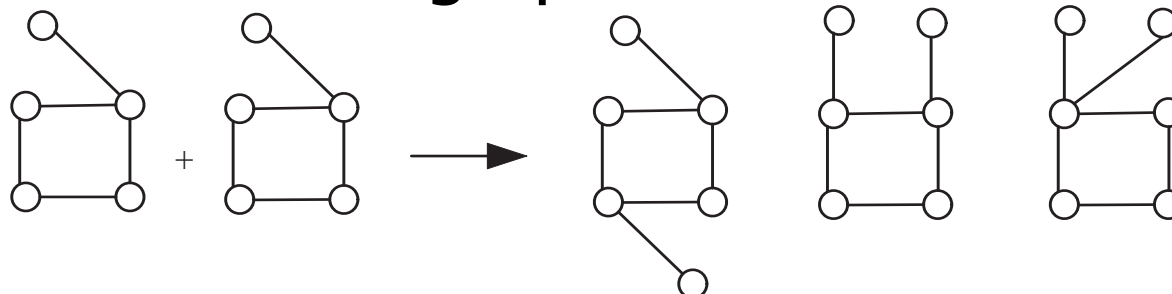
Apriori-Based, Breadth-First Search

- Methodology: breadth-search, joining two graphs



- AGM (Inokuchi, et al. PKDD'00)

- generates new graphs with one more node

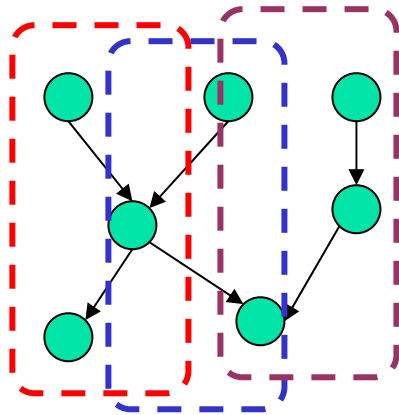


- FSG (Kuramochi and Karypis ICDM'01)

- generates new graphs with one more edge

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

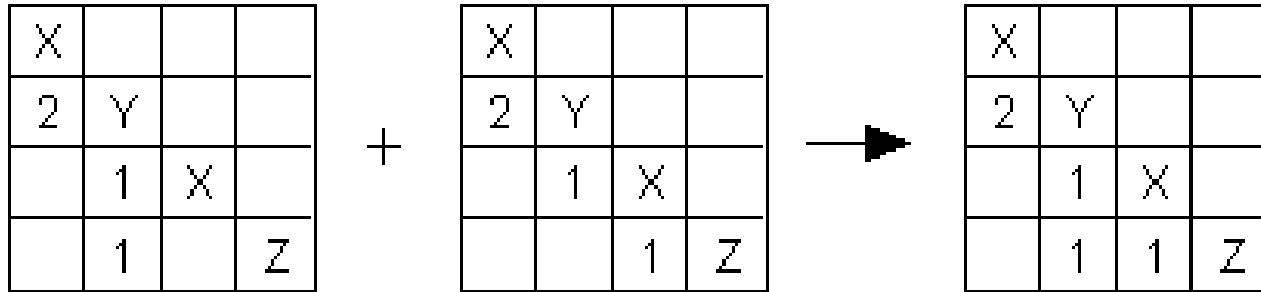
- Apriori-based approach
- Building blocks: edge-disjoint path



A graph with 3 edge-disjoint paths

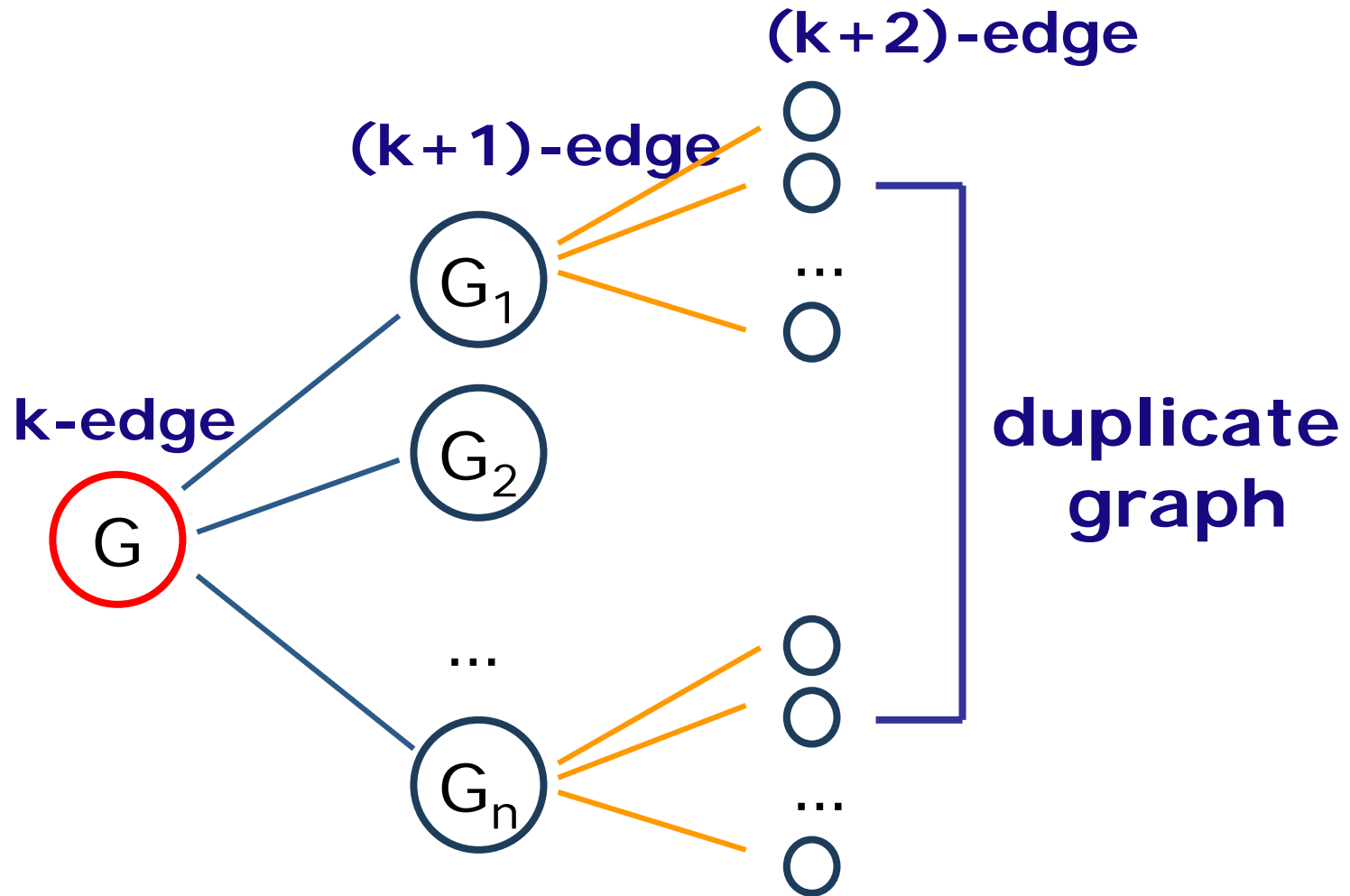
- construct frequent paths
- construct frequent graphs with 2 edge-disjoint paths
- construct graphs with $k+1$ edge-disjoint paths from graphs with k edge-disjoint paths
- repeat

FFSM (Huan, et al. ICDM'03)



- 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



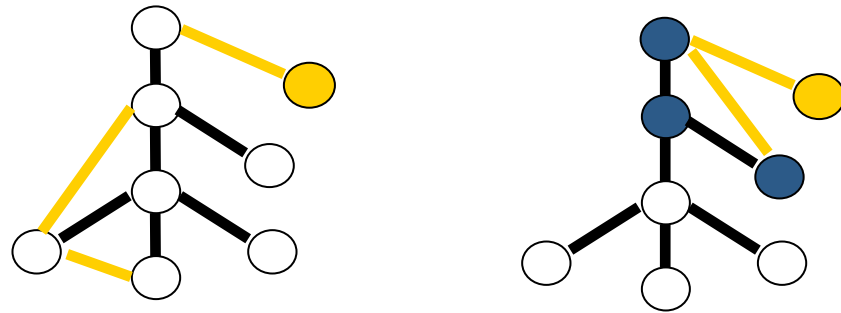


MoFa (Borgelt and Berthold ICDM'02)

- 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 FFISM and GASTON
 - Expensive Memory usage
- Local structural pruning

GSPAN (Yan and Han ICDM'02)

Right-Most Extension

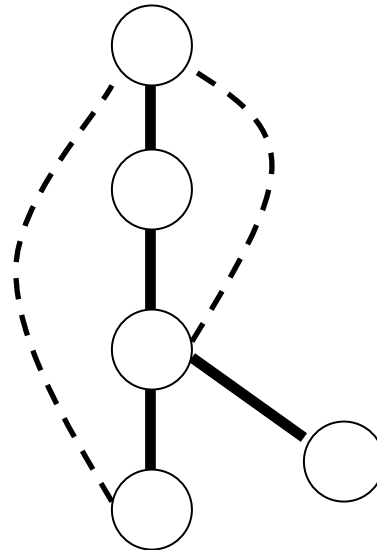
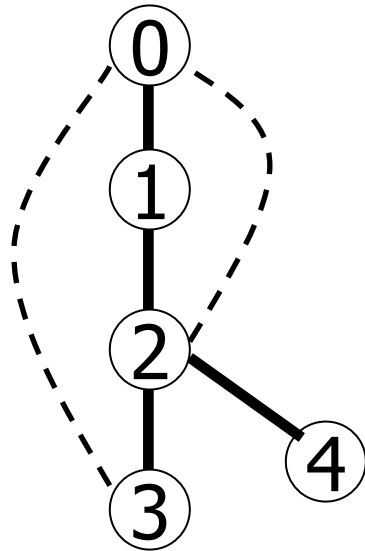


Theorem: Completeness

**The Enumeration of Graphs
using Right-most Extension is
COMPLETE**

DFS Code

- Flatten a graph into a sequence using depth first search



e0: (0,1)

e1: (1,2)

e2: (2,0)

e3: (2,3)

e4: (3,1)

e5: (2,4)

DFS Lexicographic Order

- Let **Z** be the set of DFS codes of all graphs. Two DFS codes **a** and **b** have the relation **a ≤ b** (DFS Lexicographic Order in Z) if and only if one of the following conditions is true. Let **a** = (x_0, x_1, \dots, x_n) and **b** = (y_0, y_1, \dots, y_n) ,
 - (i) if there exists **t**, $0 \leq t \leq \min(m, n)$, $x_k = y_k$ for all **k**, s.t. $k < t$, and $x_t < y_t$
 - (ii) $x_k = y_k$ for all **k**, s.t. $0 \leq k \leq m$ and $m \leq n$.

DFS Code Extension

- Let **a** be the minimum DFS code of a graph **G** and **b** be a non-minimum DFS code of **G**. For any DFS code **d** generated from **b** by one right-most extension,
 - (i) **d** is not a minimum DFS code,
 - (ii) $\text{min_dfs}(\mathbf{d})$ cannot be extended from **b**, and
 - (iii) $\text{min_dfs}(\mathbf{d})$ is either less than **a** or can be extended from **a**.

THEOREM [RIGHT-EXTENSION]

The DFS code of a graph extended from a Non-minimum DFS code is NOT MINIMUM



GASTON (Nijssen and Kok KDD'04)

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

Graph Pattern Explosion Problem

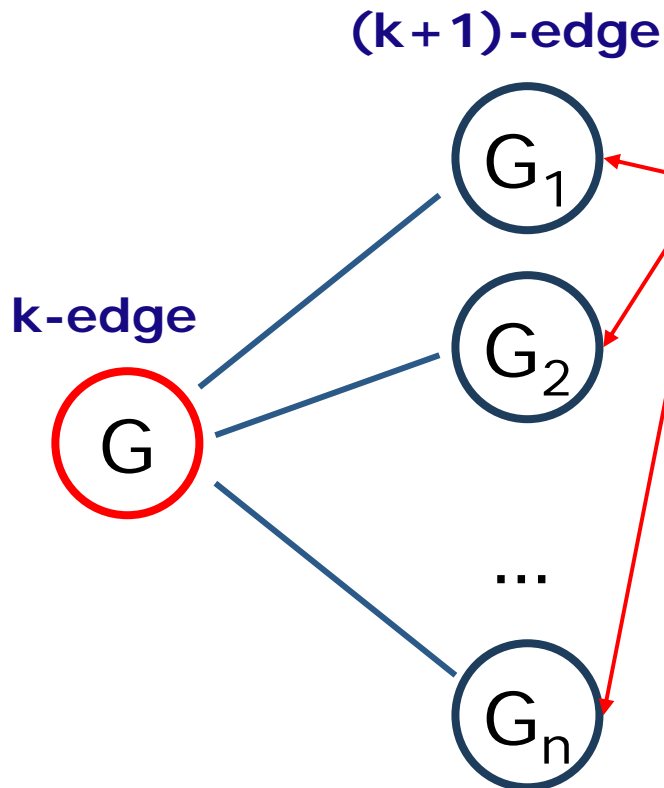
- If a graph is frequent, all of its subgraphs are frequent — **the Apriori property**
- An n -edge frequent graph may have 2^n subgraphs
- Among **422** chemical compounds which are confirmed to be active in an AIDS antiviral screen dataset, there are **1,000,000** frequent graph patterns if the minimum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

Closed Frequent Graphs

- 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 & Han, KDD'03)

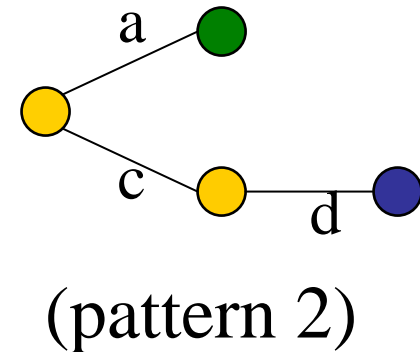
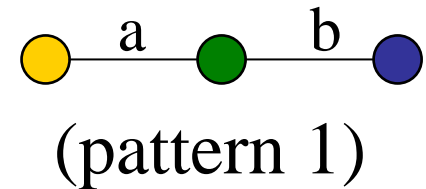
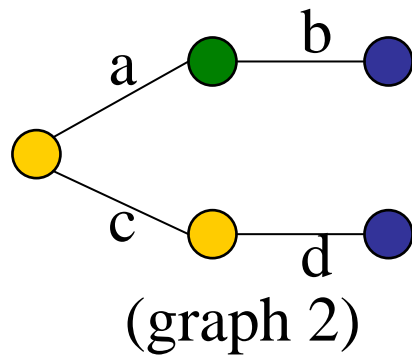
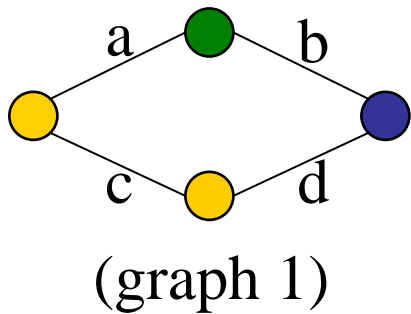
A Pattern-Growth Approach



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 the graph 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



Do the Odds Beat the Curse of Complexity?

- Potentially exponential number of frequent patterns
 - The worst case complexity vs. the expected probability
 - Ex.: Suppose Walmart has 10^4 kinds of products
 - The chance to pick up one product 10^{-4}
 - The chance to pick up a particular set of 10 products: 10^{-40}
 - What is the chance this particular set of 10 products to be frequent 10^3 times in 10^9 transactions?
- Have we solved the NP-hard problem of subgraph isomorphism testing?
 - No. But the real graphs in bio/chemistry is not so bad
 - A carbon has only 4 bounds and most proteins in a network have distinct labels

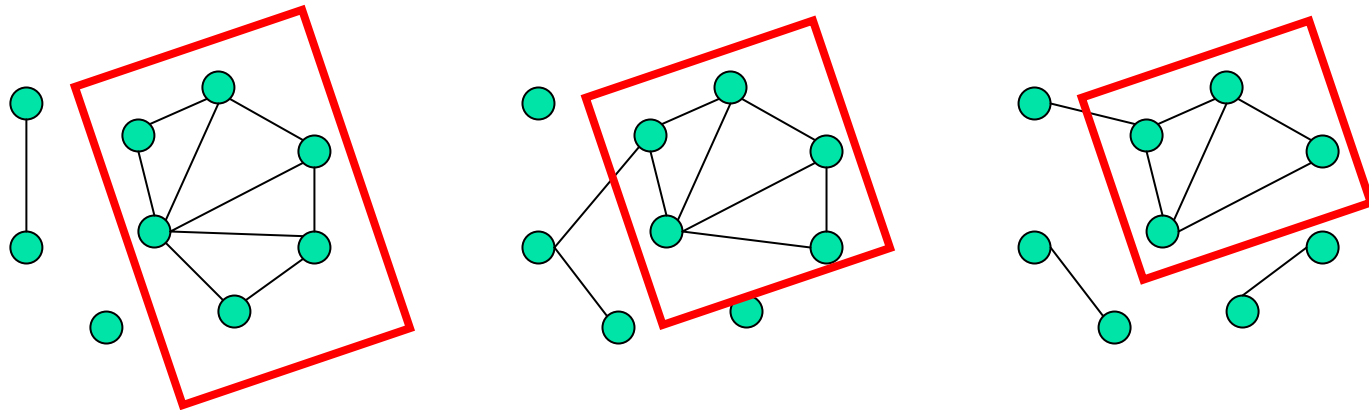


Constrained Patterns

- Density
- Diameter
- Connectivity
- Degree
- Min, Max, Avg

Constraint-Based Graph Pattern Mining

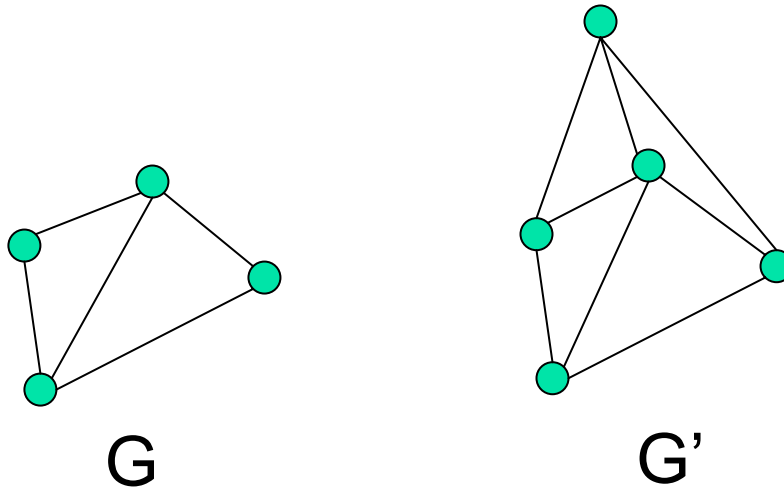
- Highly connected subgraphs in a large graph usually are not artifacts (group, functionality)



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

No Downward Closure Property

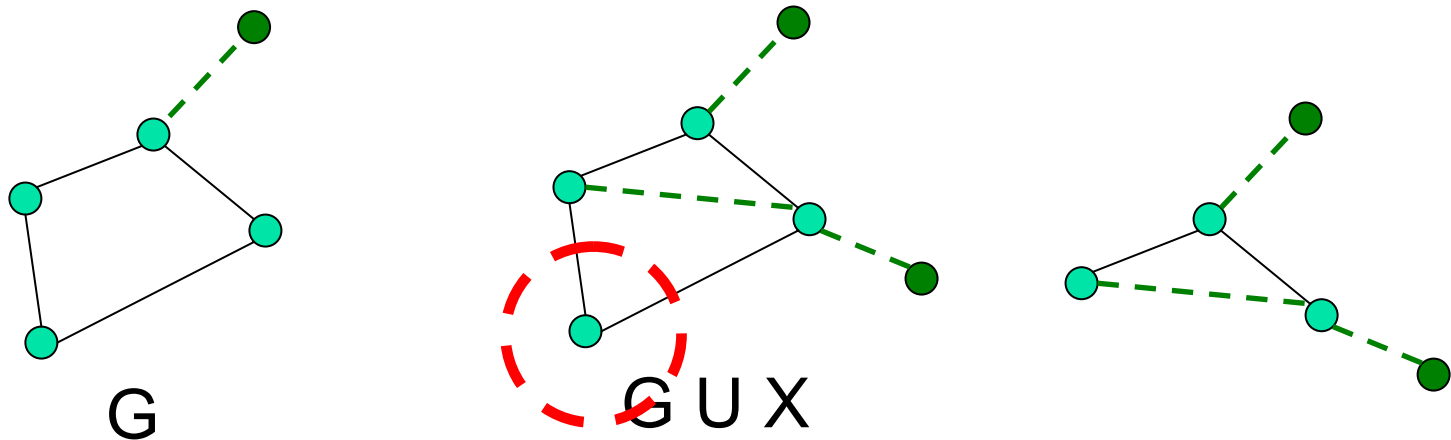
Given two graphs G and G' , if G is a subgraph of G' , it does not imply that the connectivity of G is less than that of G' , and vice versa.



Minimum Degree Constraint

Let G be a frequent graph and X be the set of edges which can be added to G such that $G \cup e$ ($e \in X$) is connected and frequent.

Graph $G \cup X$ is the maximal graph that can be Extended (one step) from the vertices belong to G



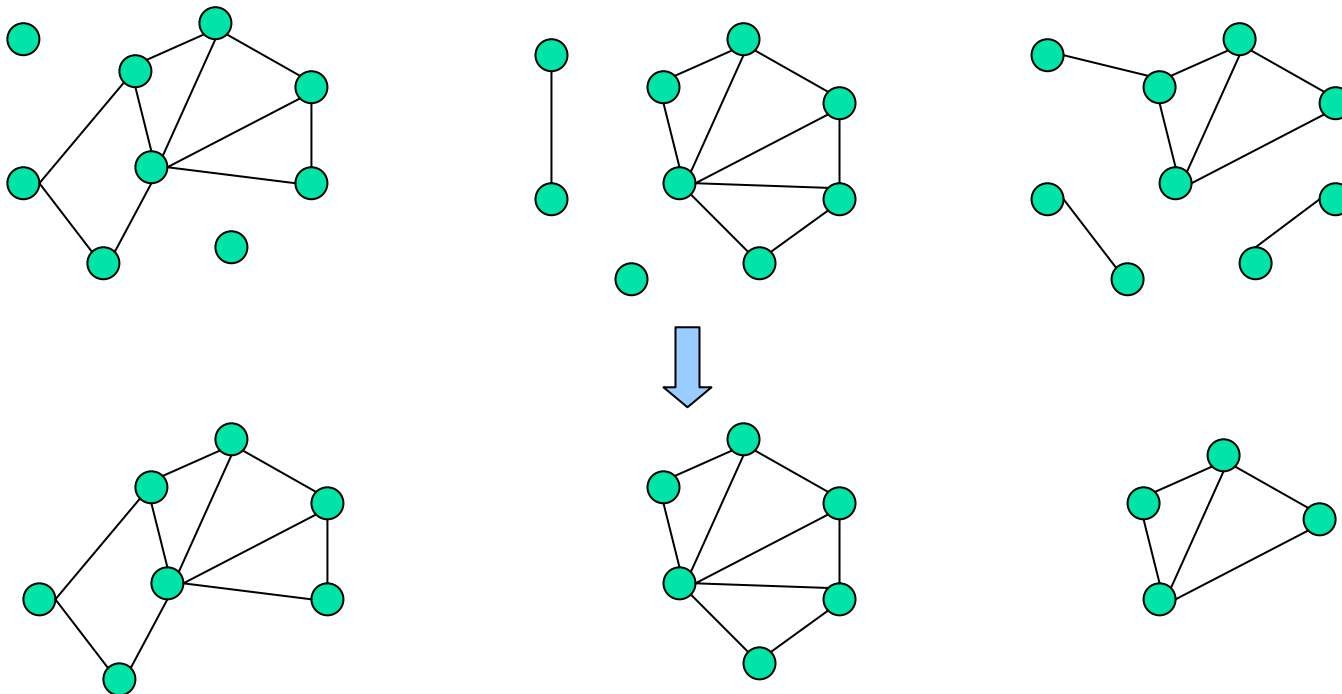


Pattern-Growth Approach

- Find a small frequent candidate graph
 - Remove vertices (shadow graph) whose degree is less than the connectivity
 - Decompose it to extract the subgraphs satisfying the connectivity constraint
 - Stop decomposing when the subgraph has been checked before
- Extend this candidate graph by adding new vertices and edges
- Repeat

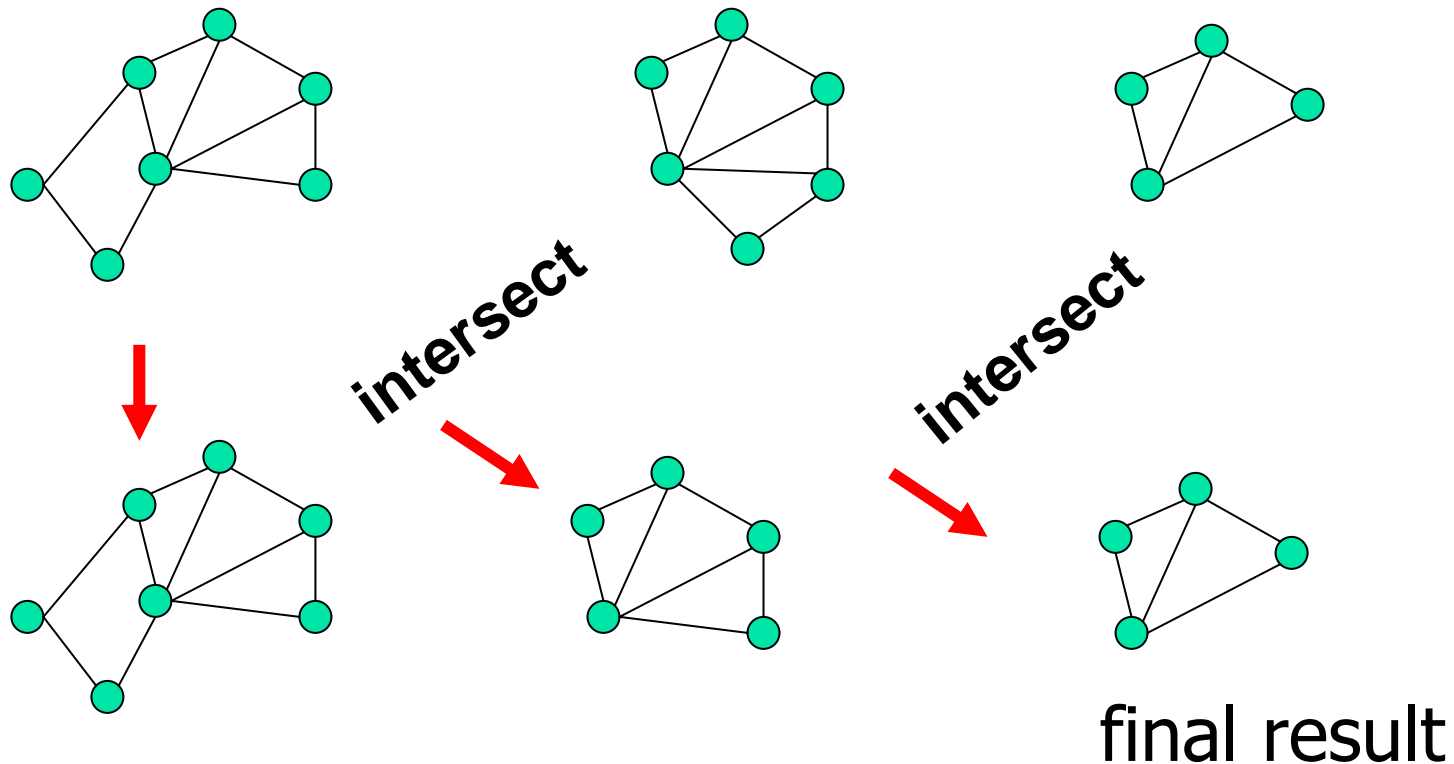
Pattern-Reduction Approach

- Decompose the relational graphs (note: no repeat nodes) according to the connectivity constraint



Pattern-Reduction Approach (cont.)

- Intersect them and decompose the resulting subgraphs





Outline

- Introduction
- Foundation
 - Graph Similarity Function
 - Graph Kernels
- Technique
 - Graph Pattern Mining
- Applications
 - Graph Indexing
 - Graph Summarization for Keyword Search



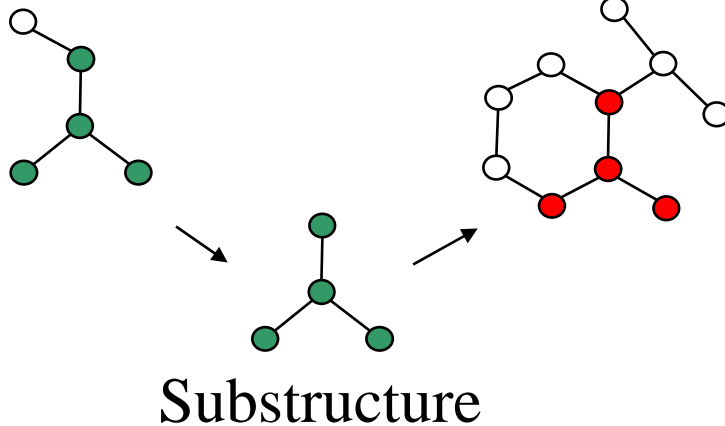
Scalability Issue

- 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

Query graph (Q)

Graph (G)



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

Cost Analysis

QUERY RESPONSE TIME

$$T_{index} + |C_q| \times (T_{io} + T_{isomorphis\ m_testing})$$

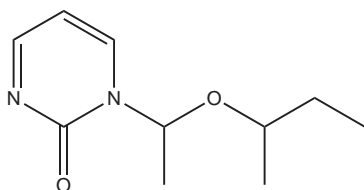
fetch index

number of candidates

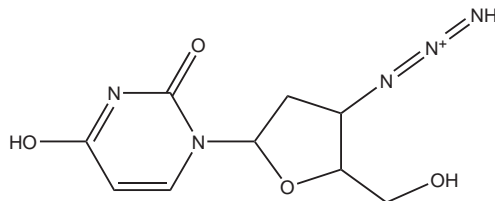
REMARK: make $|C_q|$ as small as possible

Path-based Approach

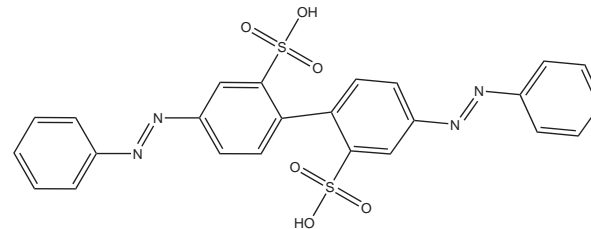
GRAPH DATABASE



(a)



(b)



(c)

PATHS

0-length: C, O, N, S

1-length: C-C, C-O, C-N, C-S, N-N, S-O

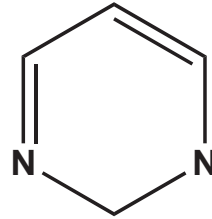
2-length: C-C-C, C-O-C, C-N-C, ...

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: $S_{C-C} = \{a, b, c\}$, $S_{C-N} = \{a, b, c\}$

2-edge: $S_{C-N-C} = \{a, b\}$, ...

...

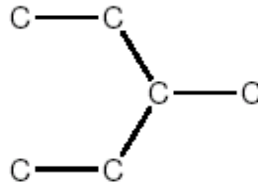
Intersect these sets, we obtain the candidate answers - graph (a) and graph (b) - which may contain this query graph.

Problems: Path-based Approach

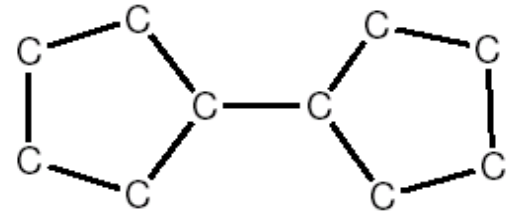
GRAPH DATABASE



(a)

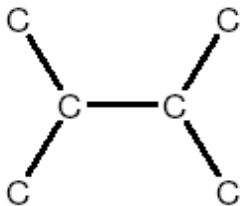


(b)



(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 graph (a) and (b).

gIndex: Indexing Graphs by Data Mining

- Our methodology on graph index:
 - Identify **frequent structures** in the database, the frequent structures are subgraphs that appear quite often in the graph database
 - Prune redundant frequent structures to maintain a small set of **discriminative structures**
 - Create an **inverted index** between

IDEAS: Indexing with Two Constraints

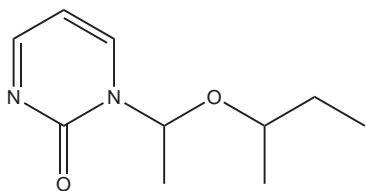
discriminative ($\sim 10^3$)

frequent ($\sim 10^5$)

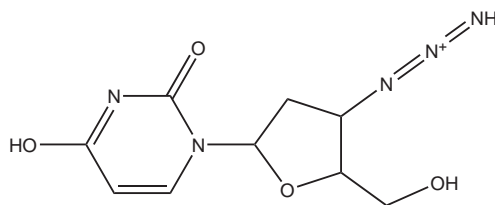
structure ($> 10^6$)

Why Discriminative Subgraphs?

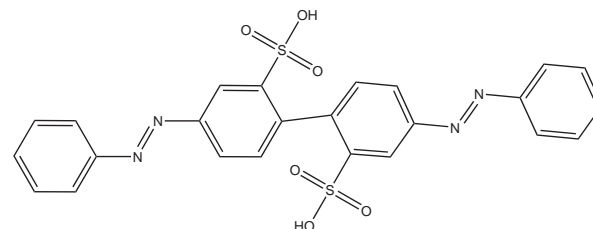
Sample database



(a)



(b)



(c)

- All graphs contain structures: C, C-C, C-C-C
- Why bother indexing these redundant frequent structures?
 - Only index structures that provide more information than existing structures

Discriminative Structures

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

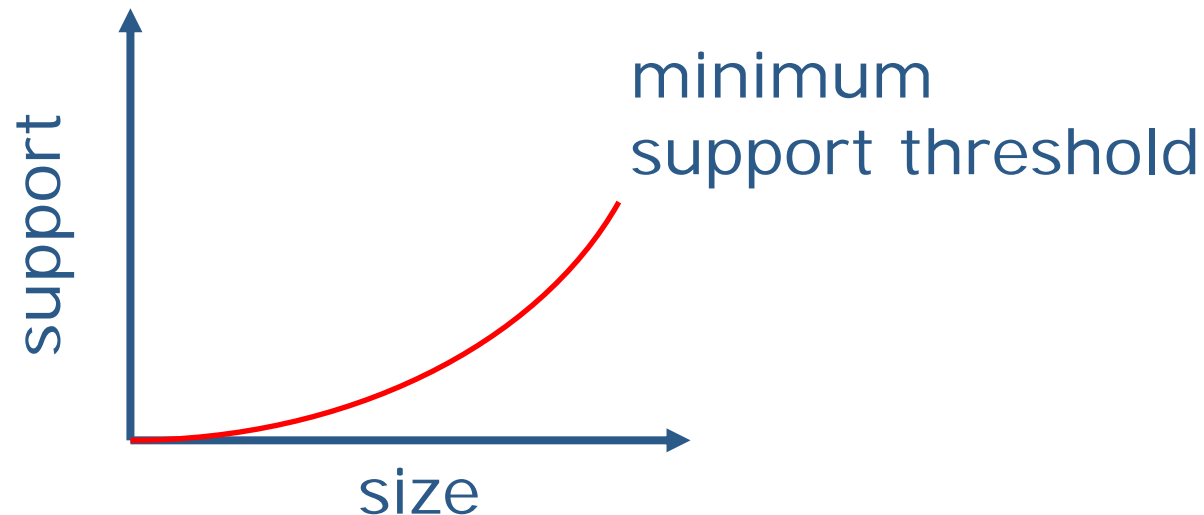
$$P(x|f_1, f_2, \dots, f_n), f_i \subset x.$$

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

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



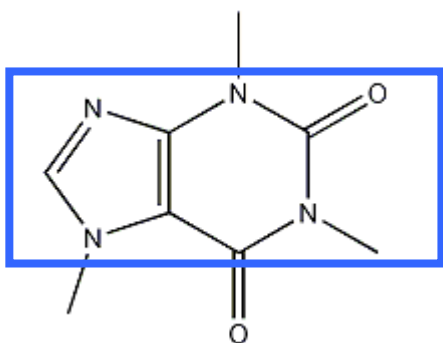
Graph Mining

- Methods for Mining Frequent Subgraphs
- Mining Variant and Constrained Substructure Patterns
- Applications:
 - Classification and Clustering
 - Graph Indexing
 - Similarity Search
- Summary

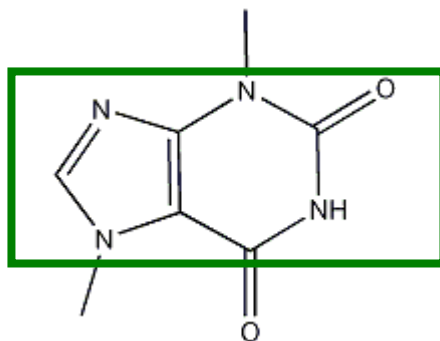


Structure Similarity Search

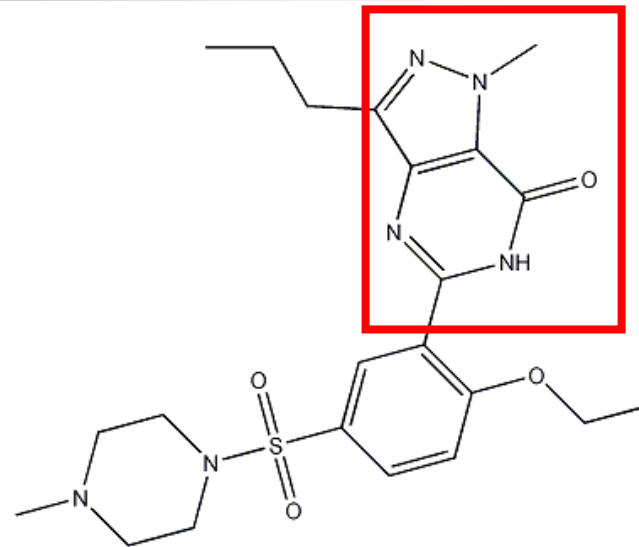
- CHEMICAL COMPOUNDS**



(a) caffeine

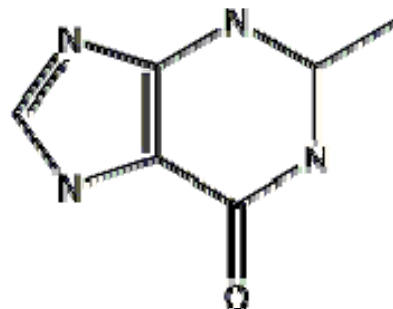


(b) diurobromine



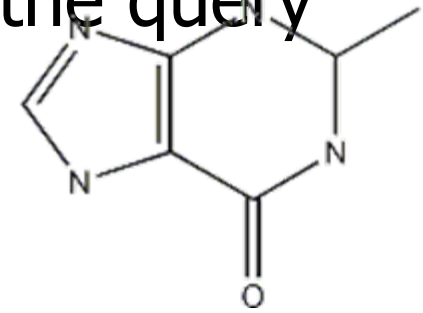
(c) viagra

- QUERY GRAPH**



Some “Straightforward” Methods

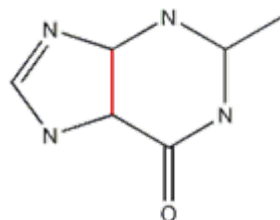
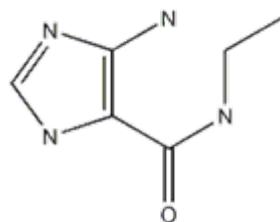
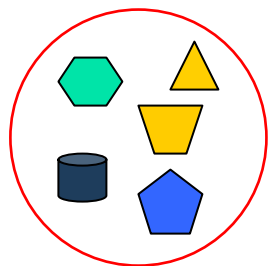
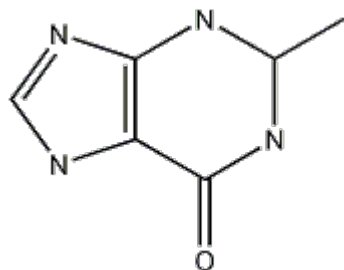
- Method 1: Directly compute the similarity between the graphs in the DB and the query graph
 - Sequential scan
 - Subgraph similarity computation
- Method 2: Form a set of subgraph queries from the original query graph and use the exact subgraph search
 - Costly: If we allow 3 edges to be missed in a 20-edge query graph, it may generate 1,140 subgraphs



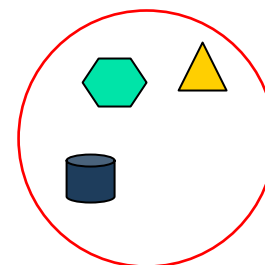
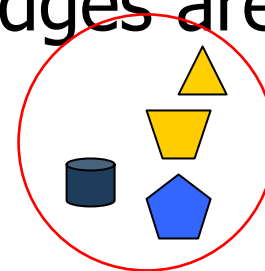
Substructure Similarity Measure

- Query relaxation measure
 - The number of edges that can be relabeled or missed; but the position of these edges are not fixed

QUERY GRAPH



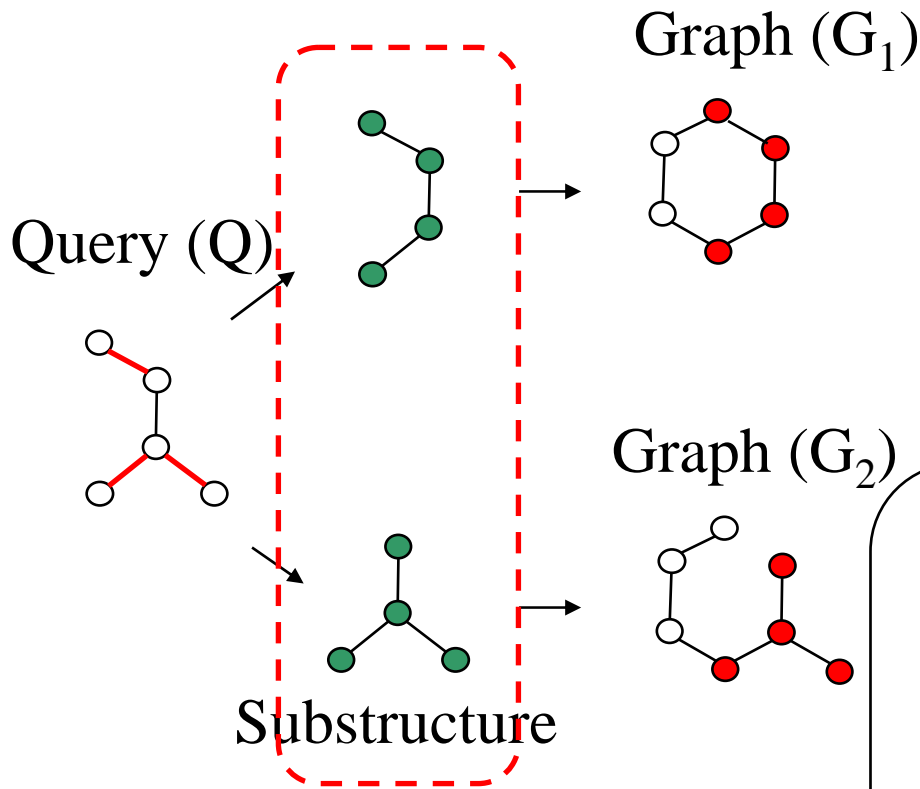
...



Substructure Similarity Measure

- **Feature-based similarity measure**
 - Each graph is represented as a feature vector $X = \{x_1, x_2, \dots, x_n\}$
 - Similarity is defined by the distance of their corresponding vectors
 - Advantages
 - Easy to index
 - Fast
 - Rough measure

Intuition: Feature-Based Similarity Search



➤ If graph **G** contains the major part of a query graph **Q**, **G** should share a number of common features with **Q**

➤ **Given a relaxation ratio, calculate the maximal number of features that can be missed !**

At least one of them should be contained

Feature-Graph Matrix

graphs in database

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 and at most 2 features to miss due to the relaxation threshold

Edge Relaxation—Feature Misses

- If we allow k edges to be relaxed, J is the maximum number of features to be hit by k edges—it becomes the maximum coverage problem

- NP-complete

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

- A greedy algorithm exists

- We design a heuristic to refine the bound of feature misses

Query Processing Framework

- Three steps in processing approximate graph queries

Step 1. Index Construction

- Select small structures as features in a graph database, and build the **feature-graph matrix** between the features and the graphs in the database

Framework (cont.)

Step 2. Feature Miss Estimation

- Determine the indexed features belonging to the query graph
- Calculate the upper bound of the number of features that can be missed for an approximate matching, denoted by J
 - On the query graph, not the graph database

Framework (cont.)

Step 3. Query Processing

- Use the feature-graph matrix to calculate the difference in the number of features between graph G and query Q , $F_G - F_Q$
- If $F_G - F_Q > J$, discard G . The remaining graphs constitute a candidate answer set



Outline

- Introduction
- Foundation
 - Graph Similarity Function
 - Graph Kernels
- Technique
 - Graph Pattern Mining
- Applications
 - Graph Indexing
 - Graph Summarization for Keyword Search



Motivation

- Keyword search is pervasive
 - For webpages, textual documents, ...
 - For relational DBs, XMLs
- Data sources are pervasive
 - Millions of Deep Web databases
 - GoogleBase, Flickr, ...
- Need to support keyword search over distributed DBs
 - Source Selection

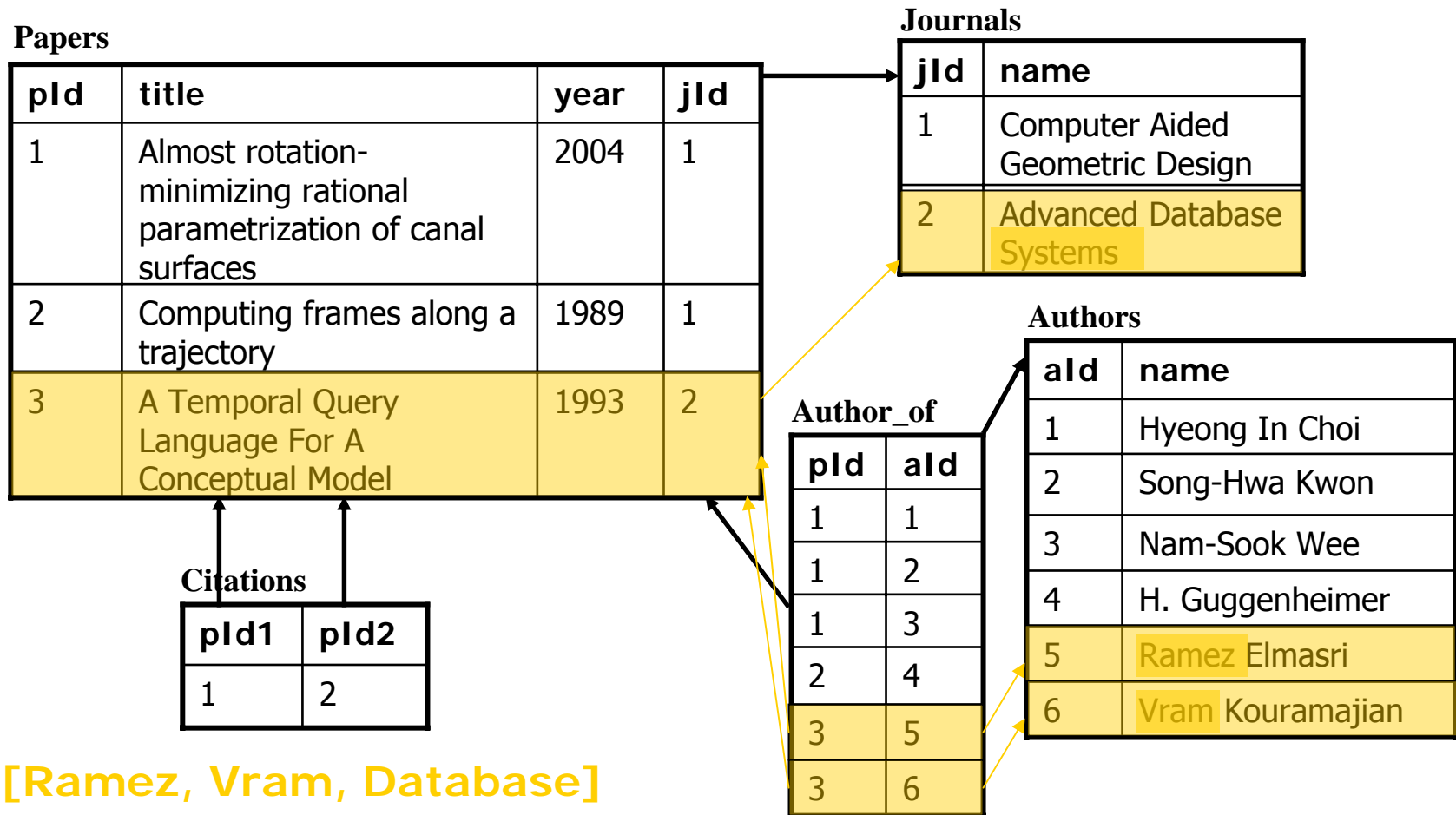


Background 1:

Keyword search over relational DBs

- The results are trees of joined tuples that contain all the query keywords
 - Graph search problem
- The search/result space is huge
 - Need good ranking method
 - Top-K processing
- Existing proposals
 - DISCOVER, BANKS, DBXplorer, ...

Example – keyword search in relational DB





DISCOVER [VLDB 2002, VLDB 2003]

- Exploits schema
 - Use FK (foreign key) references between tables
- Search steps
 - Find all **tuple sets** (TSs), subsets of tuples of native tables containing keywords
 - Find **CNs** (Candidate Networks)
 - Join expressions connecting TSs and native tables
 - Generate potential results
 - Discovered based on schema
 - Evaluate CNs
 - Top-K processing

■ Ranking

- Score of a result $T = (t_1, t_2, \dots, t_n)$

$$score(T) = \sum score(t_i) / size(T)$$

- Scores of individual tuples are based on DBMS's IR search

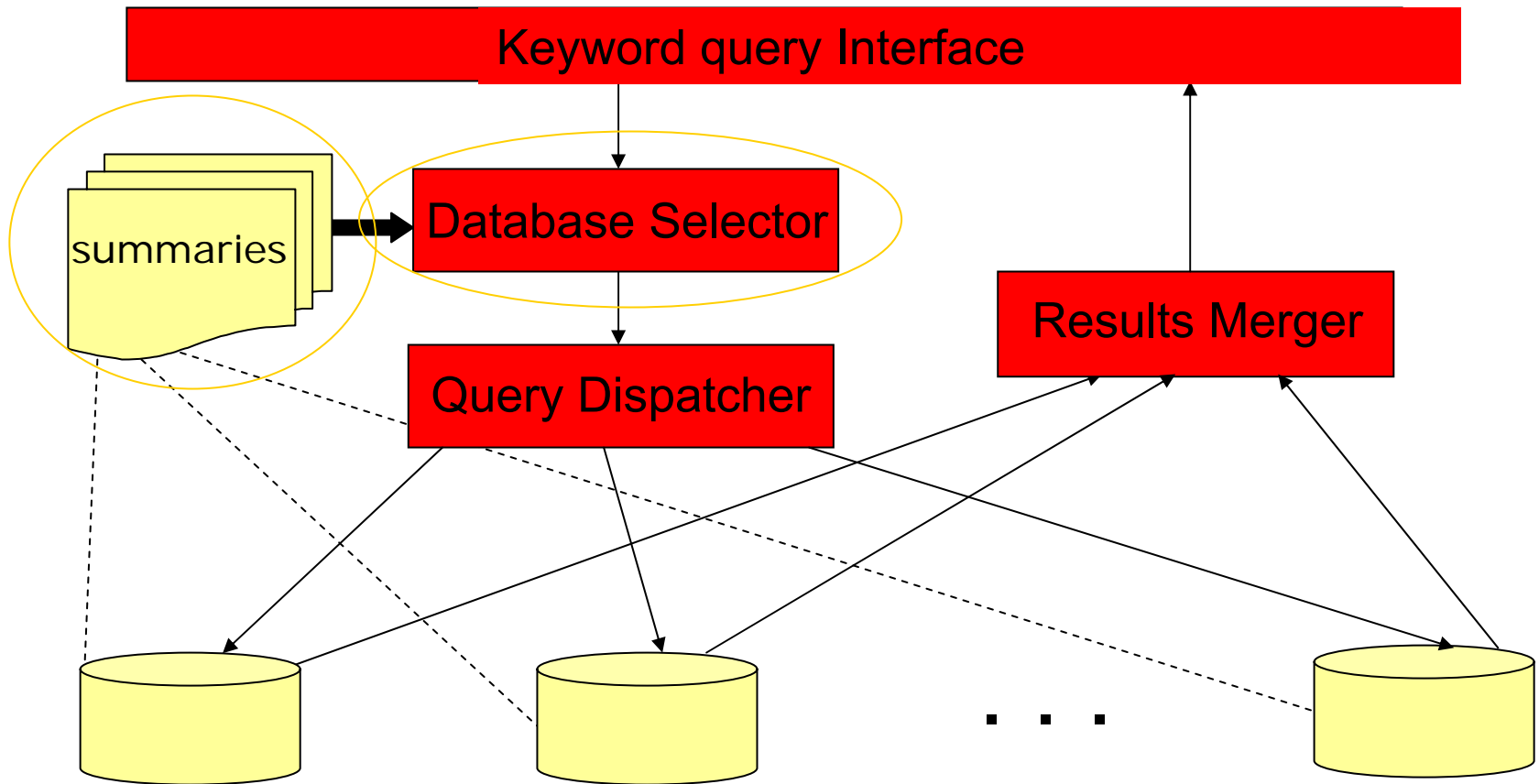
$$score(t) = \sum_{a \in t} score(a, t)$$



BANKS [ICDE 2002, VLDB 2005]

- Model the database as a graph
 - Nodes are tuples, edges are references
- Find top- K minimal Steiner trees
 - Leaves are nodes containing query keywords
- Search heuristics
 - Backward search
 - Bidirectional search
- Ranking
 - Combination of node weights and edge weights

Background2: Distributed Search Over Community Databases





Compare to Present Integrated Approach

- Just-In-Time: No pre-integration needed at a central server
- Advantages= Whatever advantages that distributed approach have over centralized approach



Build summaries for text documents

- List keywords and their frequency statistics
 - Eg. #documents containing a keyword
 - GLOSS
- May also include some inter-collection statistics
 - Eg. ICF (Inverse Collection Frequency), #sources containing a keyword
 - CORI
- Inadequate for structured database
 - Cannot capture semantics hidden behind the structure

Example – keyword list summary over RDB

Query: [multimedia, database, VLDB]

DB1

Inproceedings

id	inprocID	title	procID	year	month	annotate
t ₁	Adiba1986	Historical Multimedia Databases	23	1988	Aug	temporal
t ₂	Abarbanel 1987	Connections Perspective and Reformation	18	1987	May	Intellicorp

Conferences

id	procID	Conference
t ₃	23	the Conference on Very Large Databases (VLDB)
t ₄	18	ACM SIGMOD Conf. on the Management of Data

Keyword list summary:

DB1	Multimedia	1
	database	2
	VLDB	1

Example – keyword list summary over RDB

DB2

Query: [multimedia, database, VLDB]

titles

id	title	citKey
t ₁	A Multimedia Component Kit	Mey93a
t ₂	Managing Distributed Databases	Burl94a
t ₃	Activity Model: A Declarative Approach for Capturing Communication Behaviour in Object-Oriented Database	Liu92b
t ₄	Direct Manipulation of Temporal Structures in a Multimedia Application Framework	Acke94a

citKeywd

id	citKey	keyWdId
t ₅	Mey93a	2
t ₆	Mey93a	7
t ₇	Burl94a	302
t ₈	Liu92b	2
t ₉	Acke94a	2

keywords

id	word	keyWdId
t ₁₂	binder	2
t ₁₃	olit	7
t ₁₄	databases	302

citBkTitle

id	citKey	bkTitleId
t ₁₀	Mey93a	306
t ₁₁	Liu92b	279

booktitle

id	bkTitleNm	bkTitleId
t ₁₅	Proceedings ACM Multimedia '93	306
t ₁₆	Proceedings of the 18th VLDB Conference	279

Keyword list summary:

DB2	
multimedia	3
database	3
VLDB	1

Example – DB selection with keyword list summary

Query: [multimedia, database, VLDB]

Keyword list
summary of DB1

DB1	
Multimedia	1
database	2
VLDB	1

Keyword list
summary of DB2

DB2	
Multimedia	3
database	3
VLDB	1

~~Choose DB2 over DB1~~

Example – keyword-based selection of relational DBs

Query: [multimedia, database, VLDB]

DB1

Inproceedings

id	inprocID	title	procID	year	month	annotate
t ₁	Adiba1986	Historical Multimedia Databases	23	1988	Aug	temporal
t ₂	Abarbanel 1987	Connections Perspective and Reformation	18	1987	May	Intellicorp

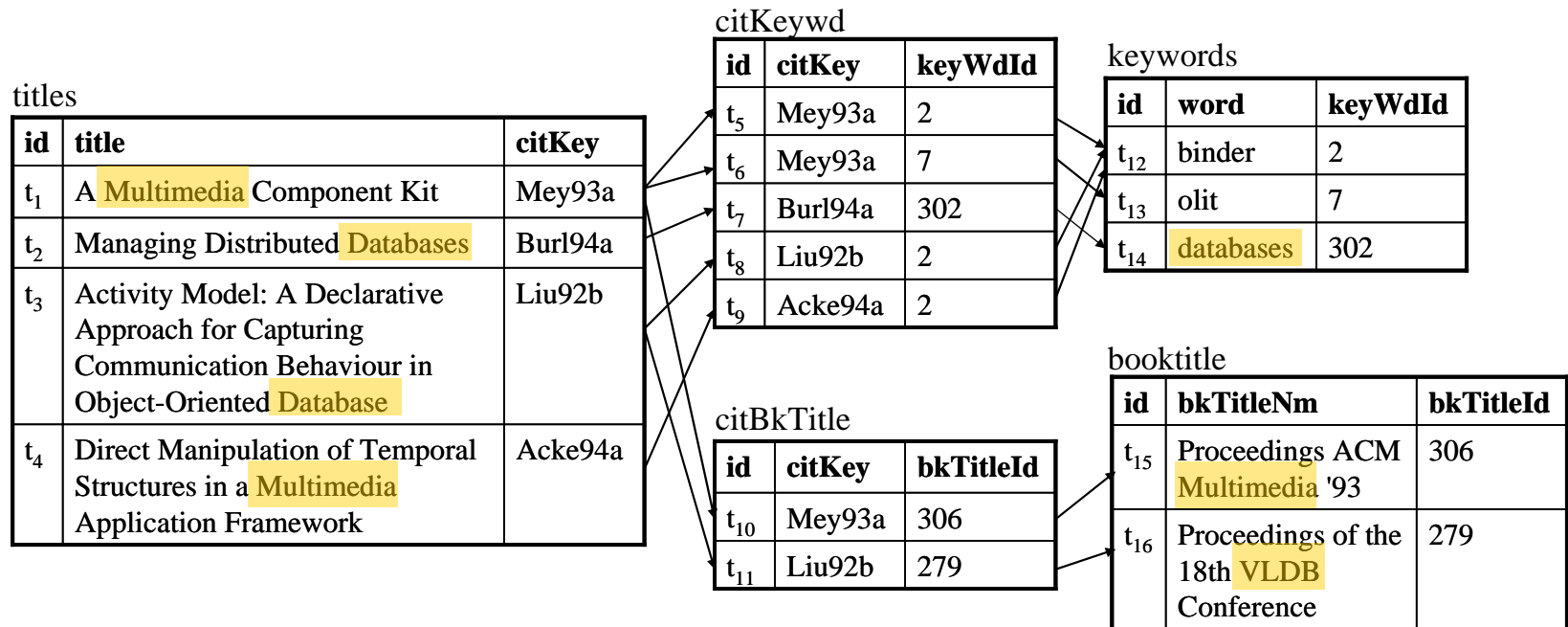
Conferences

id	procID	Conference
t ₃	23	the Conference on Very Large Databases (VLDB)
t ₄	18	ACM SIGMOD Conf. on the Management of Data

Example – keyword-based selection of relational DBs

Query: [multimedia, database, VLDB]

DB2

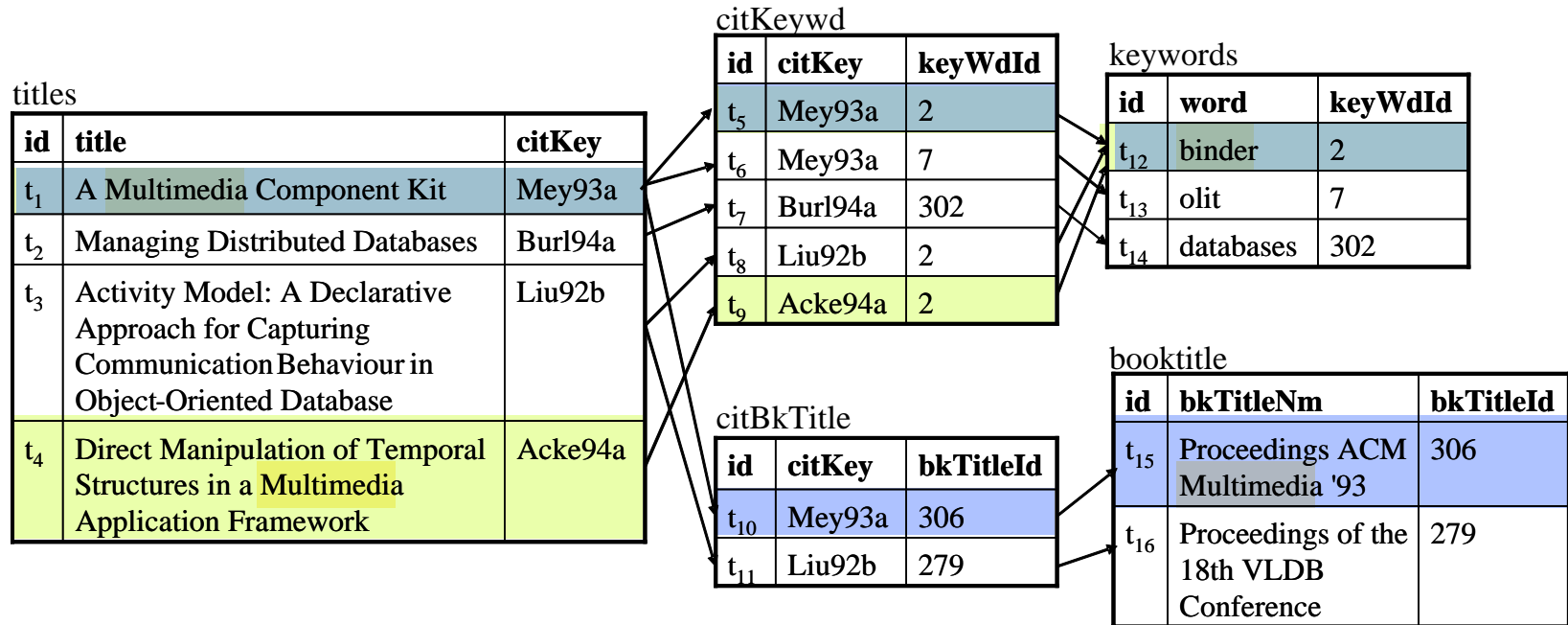


Summarizing Keyword Relationship for RDB

- Capture relationship between pairs of keywords
 - Based on connections of tuples
- Two factors
 - Distance d
 - Length of joining sequence to include two keywords
 - Frequency $\omega_d(k_i, k_j)$
 - #combinations of joining sequences at distance d

Example – distance and frequency of joining sequences

(multimedia, binder)



d = 2, freq = 2

d = 4, freq = 1

KRM: Keyword Relationship Matrix

- Models the relationship between every pair of keywords in a DB
 - $R(i, j)$ measure the goodness of top- K results given any two keywords k_i and k_j

when

$$\sum_{d=0}^{\delta} \omega_d(k_i, k_j) \leq K$$

proximity factor

$$\varphi_d = \frac{1}{d+1}$$

$$\mathcal{R}[i, j] = r_{ij} = \sum_{d=0}^{\delta} \varphi_d * \omega_d(k_i, k_j)$$

allowed maximum distance

frequency factor

the minimum distance within which there are more than K join combinations

otherwise

$$\mathcal{R}[i, j] = r_{ij} = \sum_{d=0}^{\delta'-1} \varphi_d * \omega_d(k_i, k_j) + \varphi_{\delta'} * (K - \sum_{d=0}^{\delta'-1} \omega_d(k_i, k_j))$$

Computation of KRM

- Model the content and structure of DB

$$\mathcal{D} = (d_{ij})_{m \times n} = \left\{ \begin{array}{cccc} & t_1 & t_2 & \cdots & t_n \\ k_1 & 1 & 0 & \cdots & 0 \\ k_2 & 0 & 1 & \cdots & 1 \\ \vdots & \vdots & & & \\ k_m & 1 & 0 & \cdots & 0 \end{array} \right\}$$

keywords

$$\mathcal{T} = (t_{ij})_{n \times n} = \left\{ \begin{array}{cccc} & t_1 & t_2 & \cdots & t_n \\ t_1 & 0 & 1 & \cdots & 1 \\ t_2 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & & & \\ t_n & 1 & 0 & \cdots & 0 \end{array} \right\}$$

tuples

Computation of KRM (cnt.)

- d -distance tuple relationship matrix T_d
 - Records whether there is shortest path with d hops between two tuples (1 or 0)
- $T = T_1$
- T_d can be derived inductively

$$\mathcal{T}_{d+1}[i, j] = \begin{cases} 0 & \text{if } \mathcal{T}_d^*[i, j] = 1, \\ 1 & \text{if } \mathcal{T}_d^*[i, j] = 0 \text{ and } \exists r(1 \leq r \leq n), \mathcal{T}_d[i, r] * \mathcal{T}_1[r, j] = 1 \end{cases}$$

$$\mathcal{T}_d^* = \bigvee_{k=1}^d \mathcal{T}_k$$

Computation of KRM (cnt.)

- Derive $\omega_d(k_i, k_j)$ from D and T_d

$$\mathcal{W}_0 = \mathcal{D} \times \mathcal{D}^T$$

$$\omega_0(k_i, k_j) = \mathcal{W}_0[i, j]$$

$$\mathcal{W}_d = \mathcal{D} \times \mathcal{T}_d \times \mathcal{D}^T$$

$$\omega_d(k_i, k_j) = \mathcal{W}_d[i, j]$$

- KR-summary
 - List of keyword pairs & relationship scores

Example – keyword relationship summary

Query: [multimedia, database, VLDB]

DB1

Inproceedings

id	inprocID	title	procID	year	month	annotate
t ₁	Adiba1986	Historical Multimedia Databases	23	1988	Aug	temporal
t ₂	Abarbanel 1987	Connections Perspective and Reformation	18	1987	May	Intellicorp

Conferences

id	procID	Conference
t ₃	23	the Conference on Very Large Databases (VLDB)
t ₄	18	ACM SIGMOD Conf. on the Management of Data

Frequencies of keyword pairs at difference distances

Keyword pair	d=0	d=1	d=2	d=3	d=4
database:multimedia	1	1	-	-	-
multimedia:VLDB	0	1	-	-	-
database:VLDB	1	1	-	-	-

Keyword pair	score
database:multimedia	1.5
multimedia:VLDB	0.5
database:VLDB	1.5

Example – keyword relationship summary

DB2

Query: [multimedia, database, VLDB]

titles

id	title	citKey
t ₁	A Multimedia Component Kit	Mey93a
t ₂	Managing Distributed Databases	Burl94a
t ₃	Activity Model: A Declarative Approach for Capturing Communication Behaviour in Object-Oriented Database	Liu92b
t ₄	Direct Manipulation of Temporal Structures in a Multimedia Application Framework	Acke94a

citKeywd

id	citKey	keyWdId
t ₅	Mey93a	2
t ₆	Mey93a	7
t ₇	Burl94a	302
t ₈	Liu92b	2
t ₉	Acke94a	2

keywords

id	word	keyWdId
t ₁₂	binder	2
t ₁₃	olit	7
t ₁₄	databases	302

citBkTitle

id	citKey	bkTitleId
t ₁₀	Mey93a	306
t ₁₁	Liu92b	279

booktitle

id	bkTitleNm	bkTitleId
t ₁₅	Proceedings ACM Multimedia '93	306
t ₁₆	Proceedings of the 18th VLDB	279

Keyword pair **score**

database:multimedia 0.4

multimedia:VLDB 0

database:VLDB 0.33

Frequencies of keyword pairs at difference distances

Keyword pair	d=0	d=1	d=2	d=3	d=4
database:multimedia	0	0	0	0	2
multimedia:VLDB	0	0	0	0	0
database:VLDB	0	0	1	0	0

DB Selection with KR-summary

- Estimate relationship of multiple keywords from every keyword pair's score, $rel(Q, DB)$
 - If any pair's score is 0, $rel(Q, DB) = 0$
 - Four estimations

$$rel_{min}(Q, DB) = \min_{\{k_i, k_j\} \subseteq Q, i < j} rel(k_i, k_j)$$

$$rel_{max}(Q, DB) = \max_{\{k_i, k_j\} \subseteq Q, i < j} rel(k_i, k_j)$$

$$rel_{sum}(Q, DB) = \sum_{\{k_i, k_j\} \subseteq Q, i < j} rel(k_i, k_j)$$

$$rel_{prod}(Q, DB) = \prod_{\{k_i, k_j\} \subseteq Q, i < j} rel(k_i, k_j)$$

Example – DB selection with KR-summary

Query: [multimedia, database, VLDB]

Summary of DB1

Keyword pair	score
database:multimedia	1.5
multimedia:VLDB	0.5
database:VLDB	1.5

Summary of DB2

Keyword pair	score
database:multimedia	0.4
multimedia:VLDB	0
database:VLDB	0.33

Choose DB1 over DB2





Indexing KR-Summaries

- Global index
 - Inverted lists
- Decentralized index – P2P
 - DHT, ...
 - Each peer store a subset of inverted lists



Summary

- Propose to tackle the problem of structured data source selection based on keywords
- Introduce a novel summary technique for relational DB
- Propose methods for selection of relational DBs given keyword queries
- Can more sophisticated data mining method do better?

Reference

- Hisashi Kashima, Koji Tsuda, [Akihiro Inokuchi](#): [Marginalized Kernels Between Labeled Graphs](#). ICML 2003: 321-328
- Raymond J. W. Willett P. "[Maximum common subgraph isomorphism algorithms for the matching of chemical structures](#)". Journal of Computer-Aided Molecular Design, Volume 16, Number 7, July 2002 , pp. 521-533(13)
- Bei Yu, Guoliang Li, Karen Sollins, Anthony K. H. Tung. [Effective Keyword-based Selection of Relational Databases](#). **ACM SIGMOD 2007**.
- X. Yan and J. Han, "[CloseGraph: Mining Closed Frequent Graph Patterns](#)", Proc. 2003 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'03), Washington, D.C., Aug. 2003..
- X. Yan, P. S. Yu, and J. Han, "[Substructure Similarity Search in Graph Databases](#)", in Proc. 2005 ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD'05), Baltimore, Maryland, June 2005
- HK06: Chapter 9.1

Optional References:

- H Bunke. [On a relation between graph edit distance and maximum common subgraph](#). Pattern Recognition Letters, 1997
- H Bunke, K Shearer. [A graph distance metric based on the maximal common subgraph](#). Pattern Recognition Letters, 1998 - ece.concordia.ca
- X. Yan, P. S. Yu, and J. Han, "[Substructure Similarity Search in Graph Databases](#)", in Proc. 2005 ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD'05), Baltimore, Maryland, June 2005.