Frequent Subgraph Mining (FSM)

.

Introduction

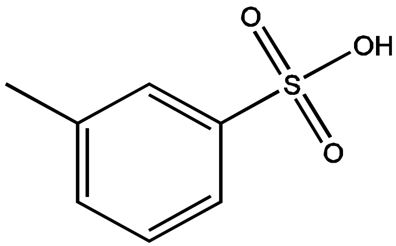
- *Frequent* subgraphs
  - A (sub)graph is ***frequent*** if its *support* (occurrence frequency) in a given dataset is no less than a *minimum support* threshold
- Applications of graph pattern mining
  - Mining biochemical structures
  - Program control flow analysis
  - Mining XML structures or Web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

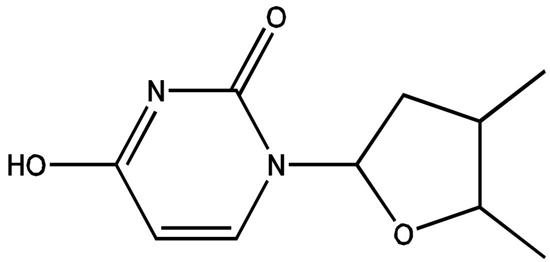
What Makes FSM So Hard?

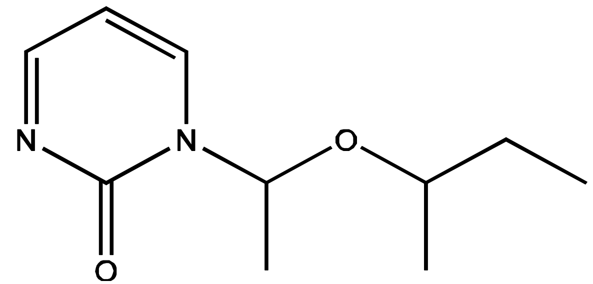
- Isomorphic graphs have same structural properties even though they may look different.
- Subgraph isomorphism problem: Does a graph contain a subgraph isomorphic to another graph?
- FSM algorithms encounter this problem while buildings graphs.
- This problem is known to be NP-complete!

Example: Frequent Subgraphs

GRAPH DATASET



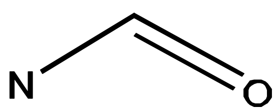




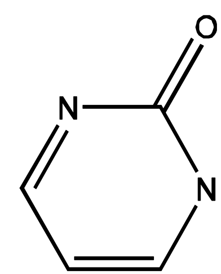
(A) (B) (C)

FREQUENT PATTERNS (MIN SUPPORT IS 2)

(1)

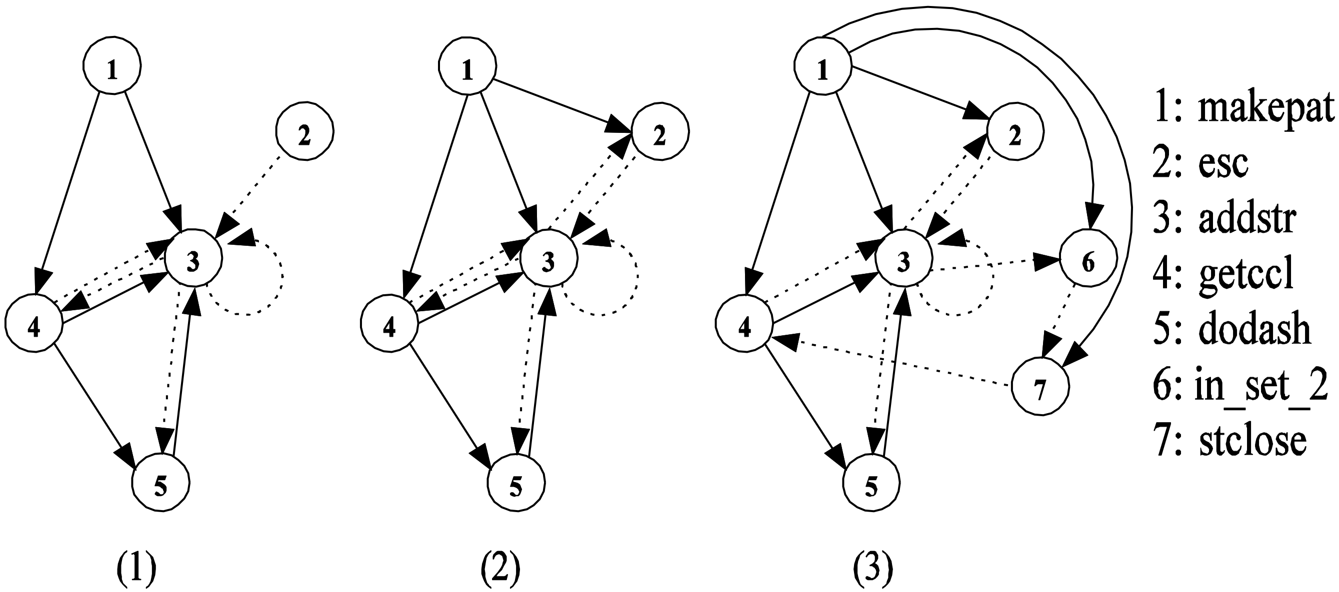


(2)

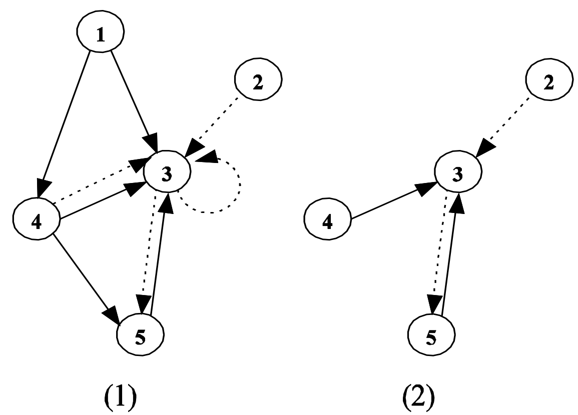


EXAMPLE (II)

GRAPH DATASET



FREQUENT PATTERNS (MIN SUPPORT IS 2)



Mining Frequent Subgraphs

- **Methods for Mining Frequent Subgraphs**
- **Applications:**
  - **Graph Indexing**
  - **Similarity Search**
  - **Classification and Clustering**
- **Summary**

Frequent Subgraphs Mining Algorithms

- Underlying strategy of both traditional frequent pattern mining and frequent subgraph mining
- General Process:
  – candidate generation: which patterns will be considered? For FSM,
  – candidate pruning: if a candidate is not a viable frequent pattern, can we exploit the pattern to prevent unnecessary work?
    • subgraphs and subsets exponentiate as size increases!
  – support counting: how many of a given pattern exist?
- These algorithms work in a breadth-first or depth-first way.
  – Joins smaller frequent sets into larger ones.
  – Checks the frequency of larger sets.

Frequent Subgraphs Mining Algorithms

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

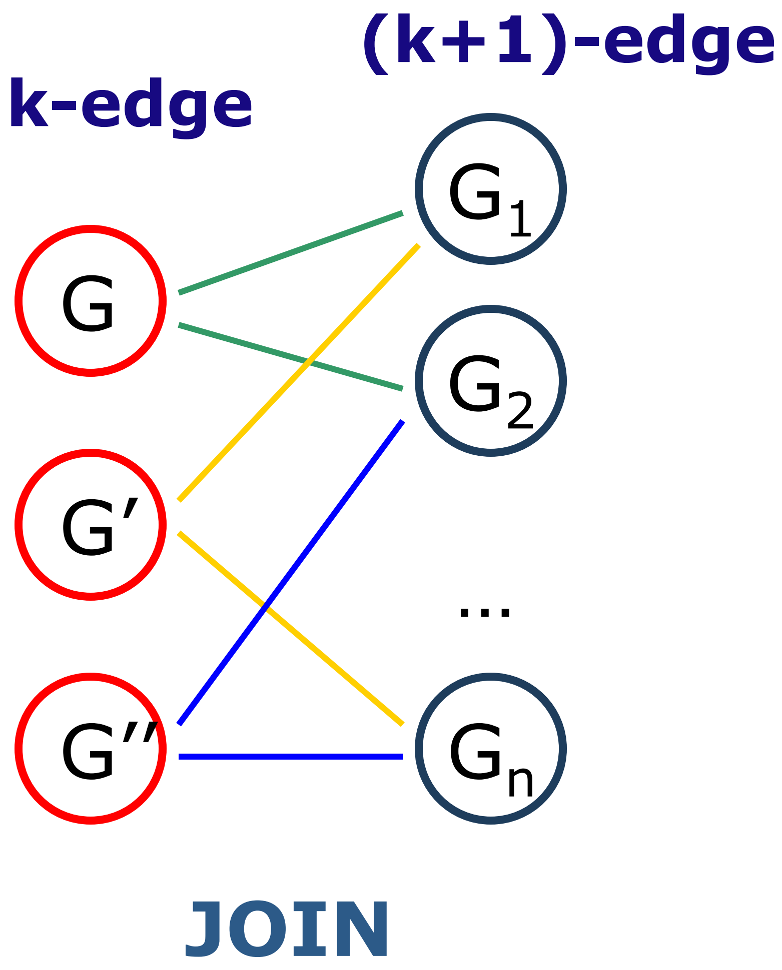
Frequent Subgraph Mining Approaches

- 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

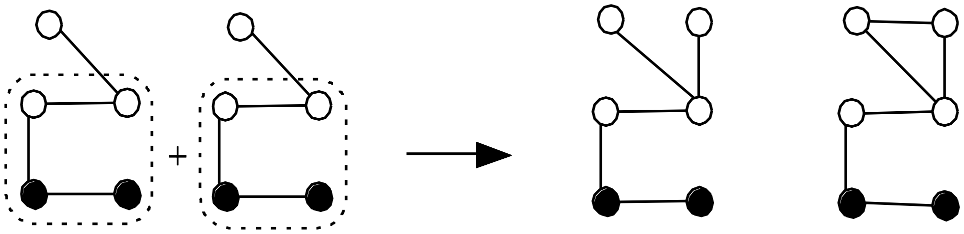
- Search order
  - breadth vs. depth
- Generation of candidate subgraphs
  - apriori vs. pattern growth
- Elimination of duplicate subgraphs
  - passive vs. active
- Support calculation
  - embedding store or not
- Discover order of patterns
  - path → tree → graph

Apriori-Based Approach

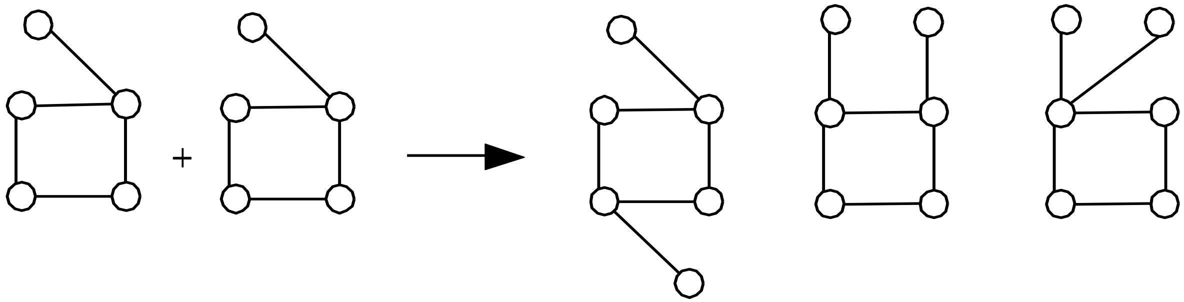


Apriori-Based, Breadth-First Search

- 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

FFSM (Huan, et al. ICDM'03)

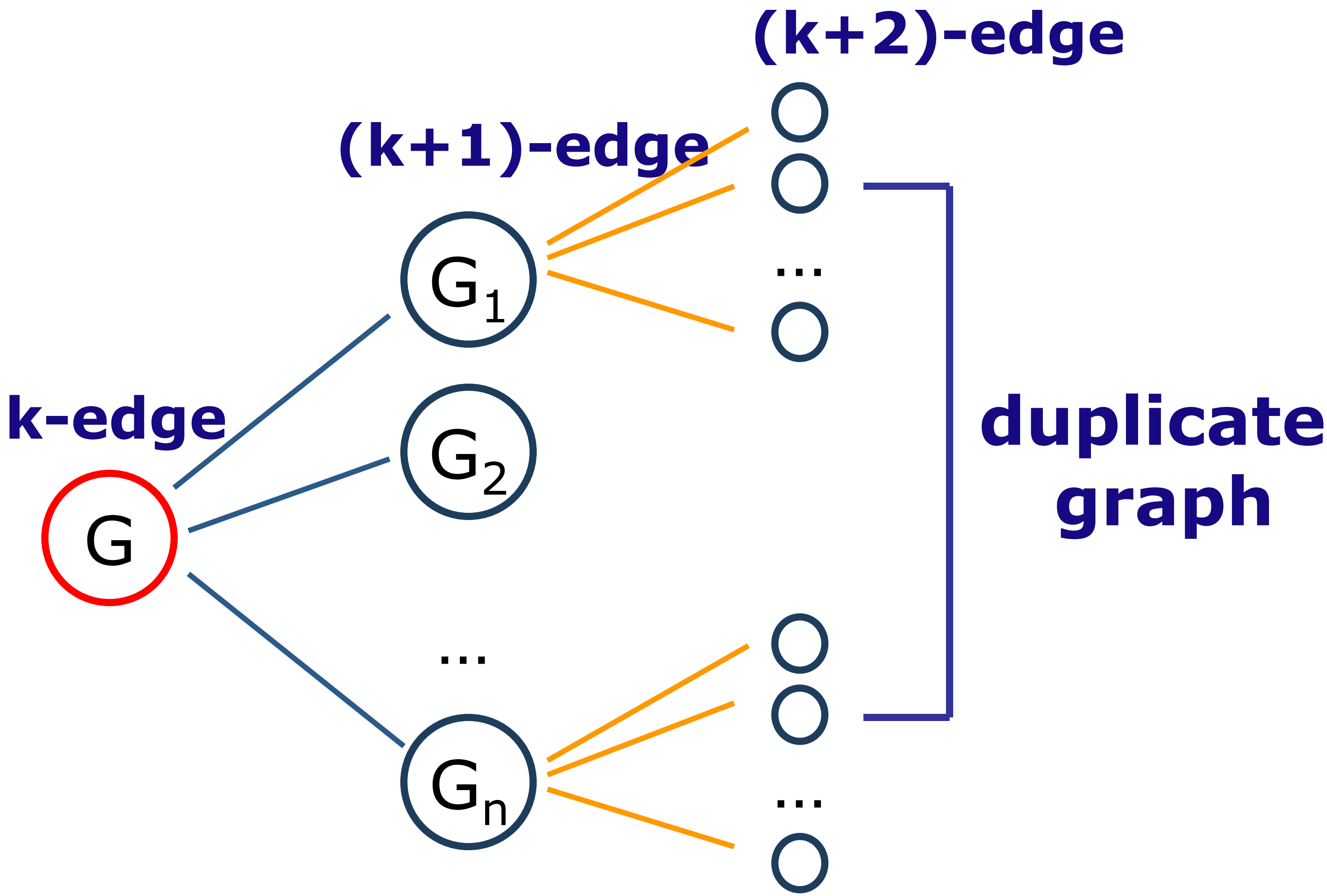
$$\begin{bmatrix} X & & & \\ 2 & Y & & \\ & 1 & X & \\ & 1 & & Z \end{bmatrix} + \begin{bmatrix} X & & & \\ 2 & Y & & \\ & 1 & X & \\ & & 1 & Z \end{bmatrix} \rightarrow \begin{bmatrix} X & & & \\ 2 & Y & & \\ & 1 & X & \\ & 1 & 1 & Z \end{bmatrix}$$

- 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

Graph Pattern Explosion Problem

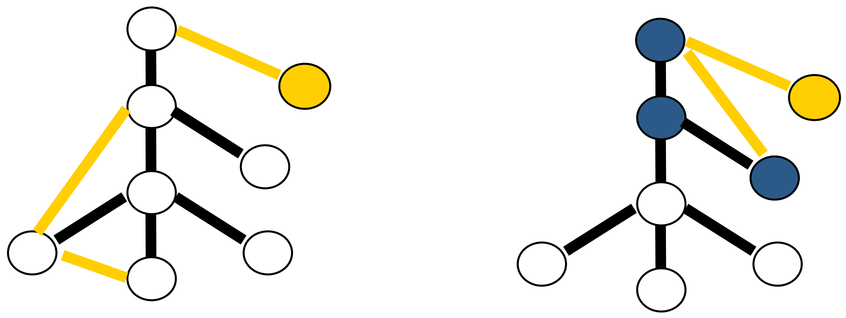
- If a graph is frequent, all of its subgraphs are frequent ─ **the Apriori property**
- An **n**-edge frequent graph may have 2^n subgraphs
- Among **422** chemical compounds which are confirmed to be active in an AIDS antiviral screen dataset, there are **1,000,000** frequent graph patterns if the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

Pattern Growth Method



GSPAN (Yan and Han ICDM'02)

Right-Most Extension

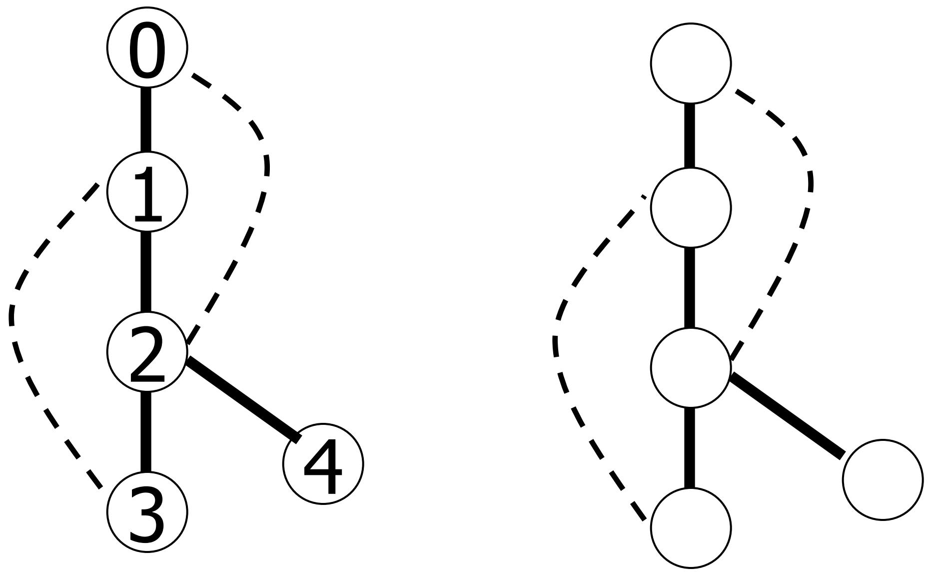


Theorem: Completeness

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 $= (x_0, x_1, \ldots, x_n)$ and

  b $= (y_0, y_1, \ldots, y_n)$,

(i) if there exists t, $0 <= t <= \min(m,n)$, $x_k = y_k$ for all k, s.t. $k < t$, and $x_t < y_t$

(ii) $x_k = y_k$ for all k, s.t. $0 <= k <= m$ and $m <= n$.

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) min_dfs(**d**) cannot be extended from **b**, and

  (iii) min_dfs(**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

Pattern-Growth Approach

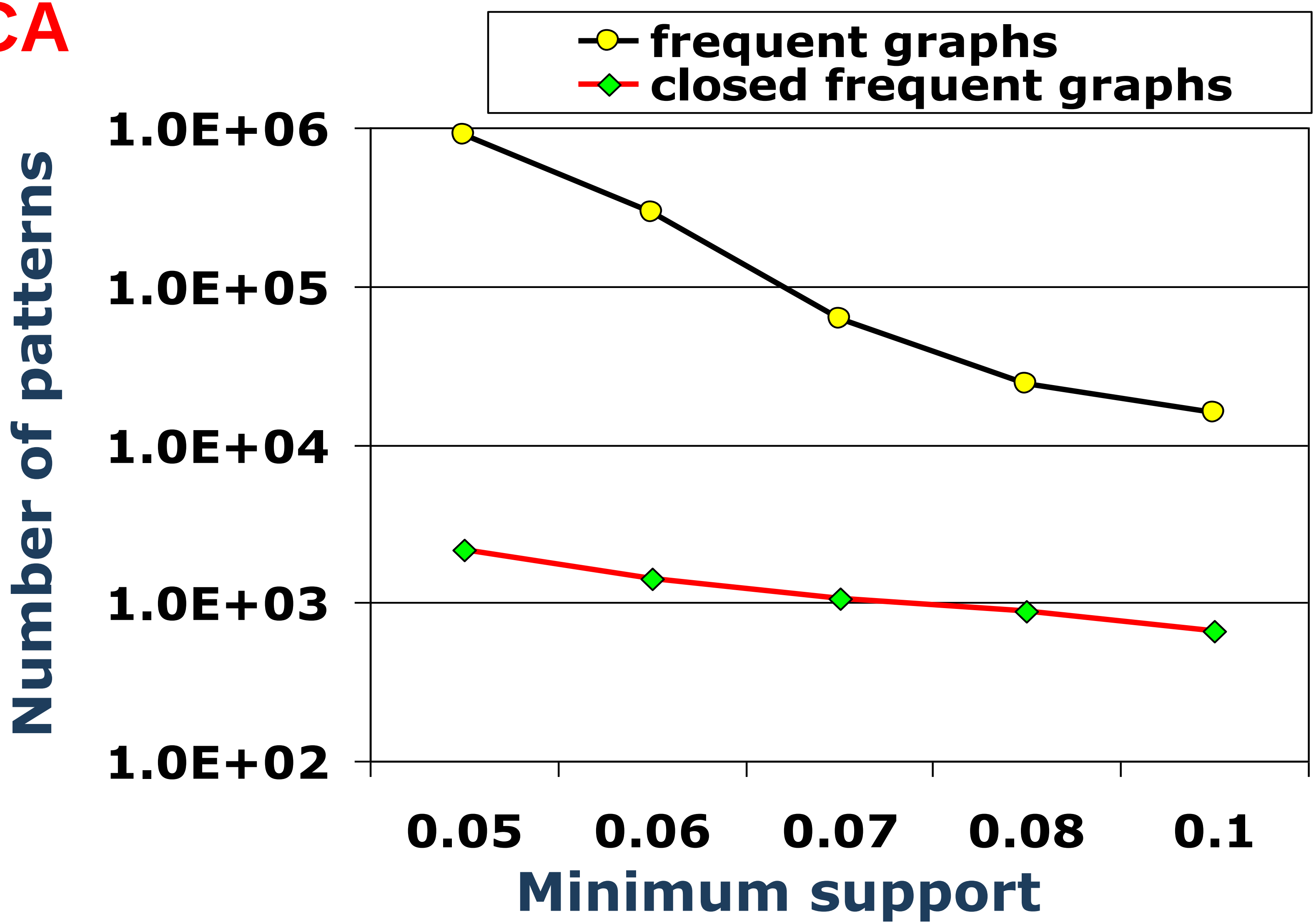
- Find a small frequent candidate graph
  - Remove vertices (shadow graph) whose degree is less than the connectivity
  - Decompose it to extract the subgraphs satisfying the connectivity constraint
  - Stop decomposing when the subgraph has been checked before
- Extend this candidate graph by adding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

Closed Frequent Graphs

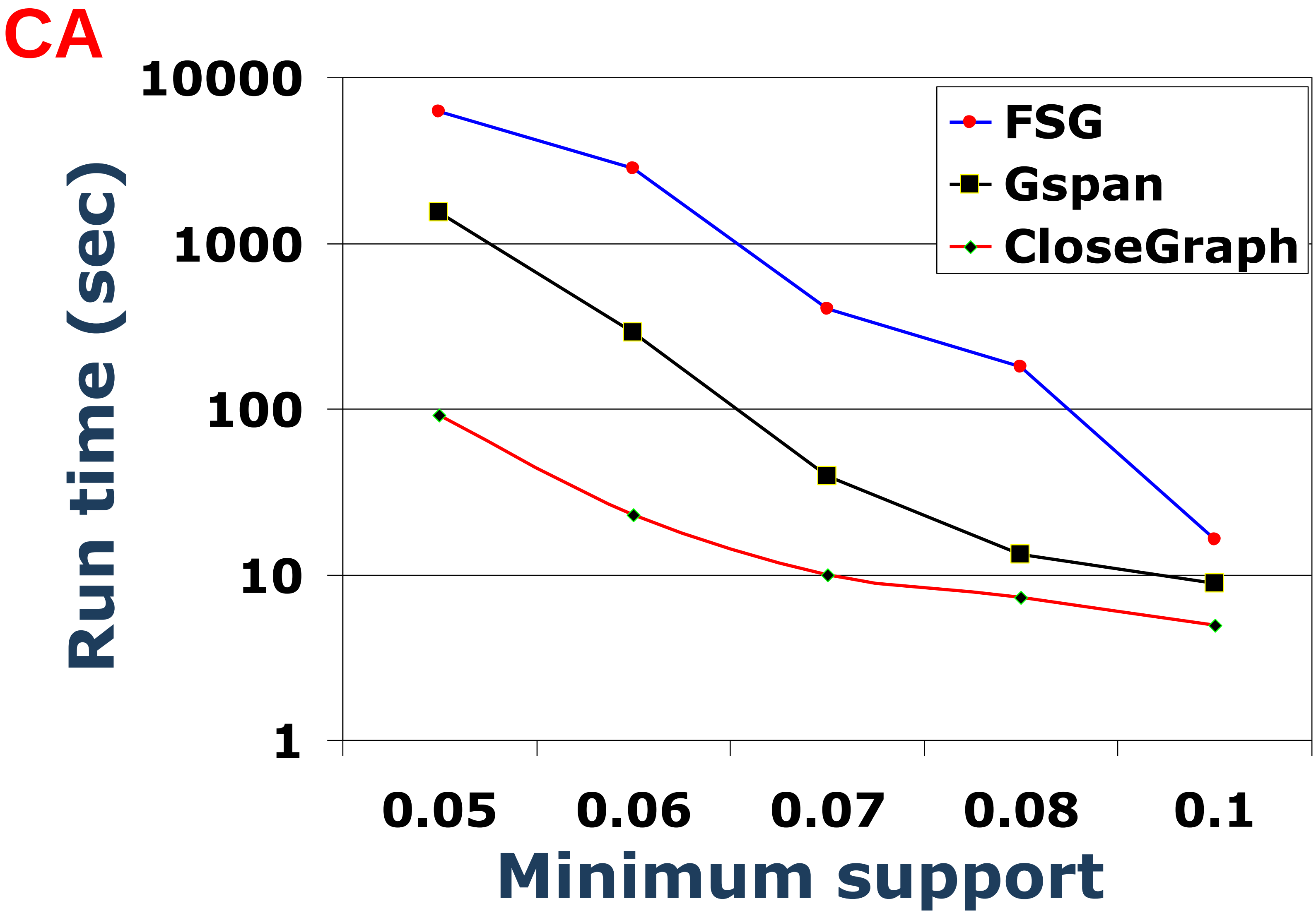
- Motivation: Handling graph pattern explosion problem
- Closed frequent graph
  - A frequent graph G is *closed* if there exists no supergraph of G that carries the same support as G
- If some of G's subgraphs have the same support, it is unnecessary to output these subgraphs (nonclosed graphs)
- *Lossless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

Number of Patterns: Frequent vs. Closed

CA



Runtime: Frequent vs. Closed



Graph Mining

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

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

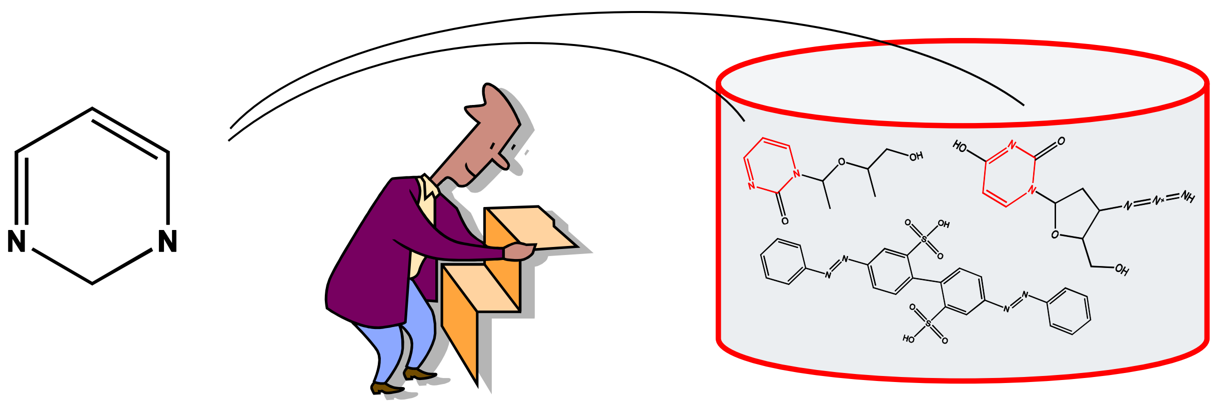
- Subgraph patterns from domain knowledge
  - Molecular descriptors
- Subgraph patterns from data mining
- General idea
  - Each graph is represented as a feature vector **x** = $\{x_1, x_2, \ldots, x_n\}$, where x_i is the frequency of the i-th pattern in that graph
  - Each vector is associated with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

Graph Mining

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

Graph Search

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

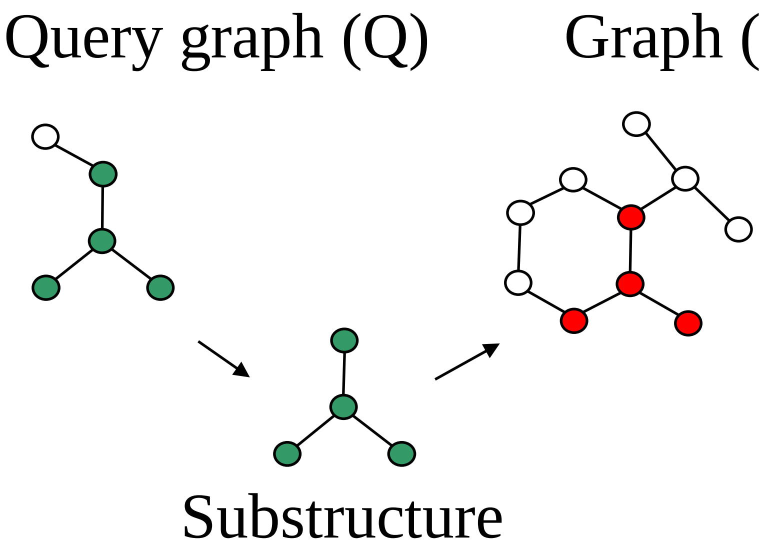


query graph **graph database**

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

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

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

fetch index

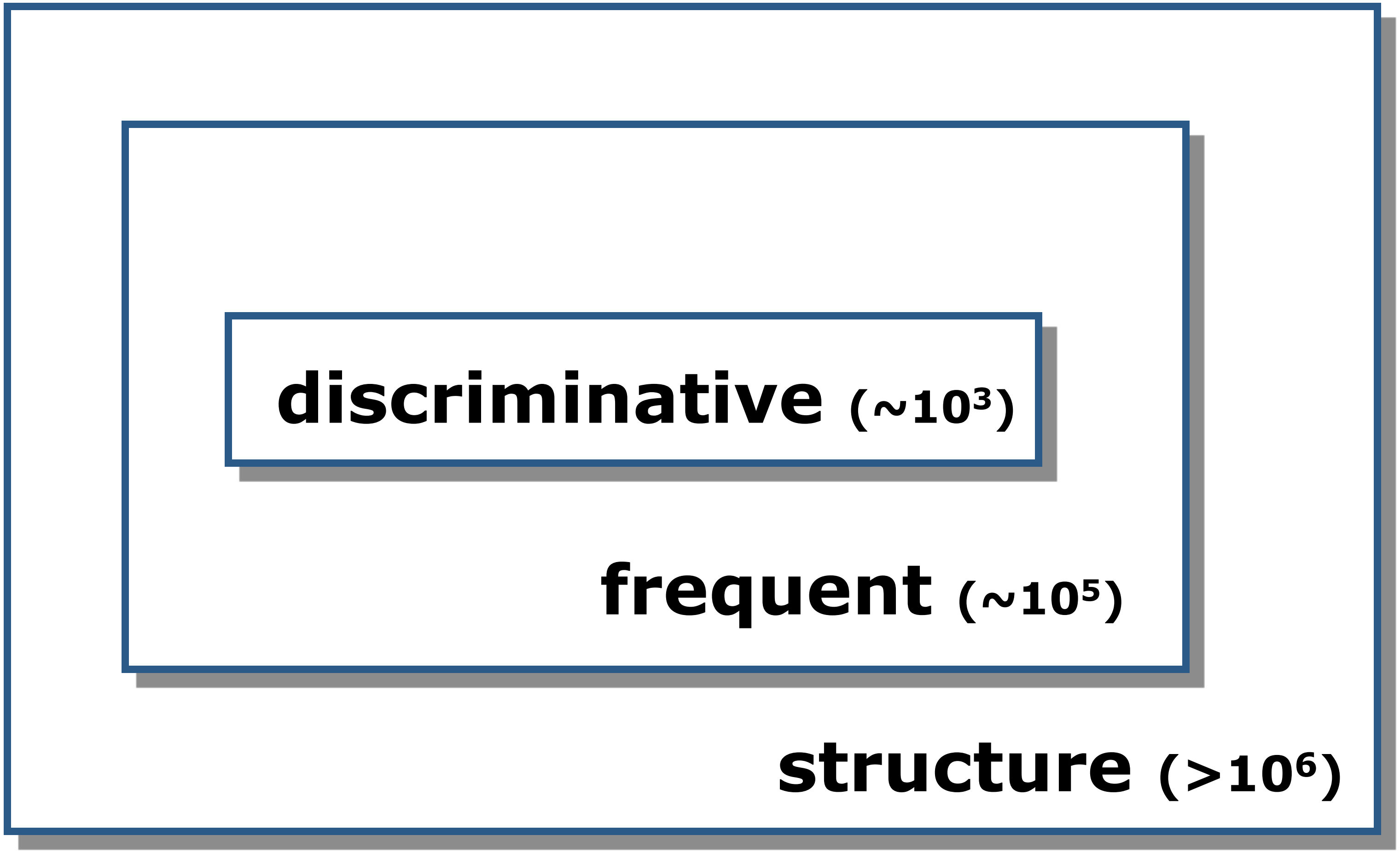
number of candidates

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

gIndex: Indexing Graphs by Data Mining

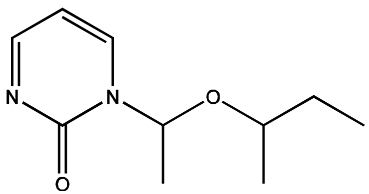
- Our methodology on graph index:
  - Identify frequent structures in the database, the frequent structures are subgraphs that appear quite often in the graph database
  - Prune redundant frequent structures to maintain a small set of discriminative structures
  - Create an inverted index between discriminative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

IDEAS: Indexing with Two Constraints

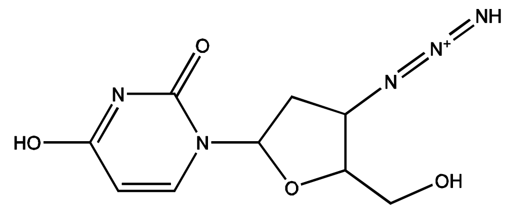


Why Discriminative Subgraphs?

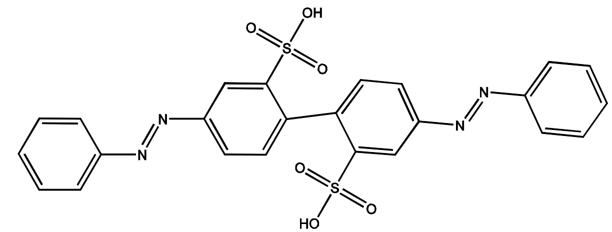
Sample database



(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

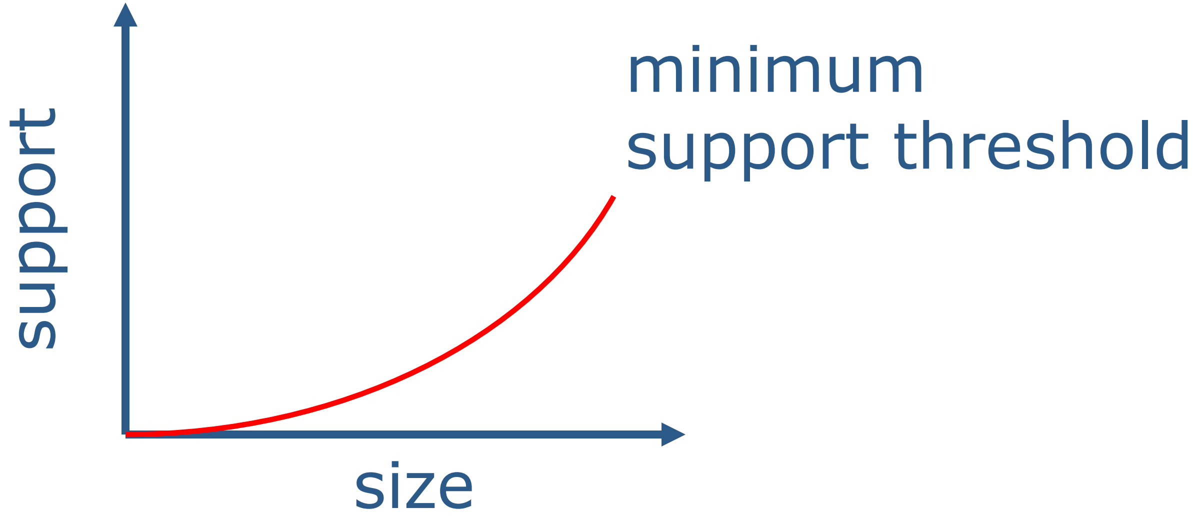
- Pinpoint the most useful frequent structures
  - Given a set of structures f1, f2,…fn and a new structure x, we measure the extra indexing power provided by x,

    P (x/ f1, f2,…fn)

    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?

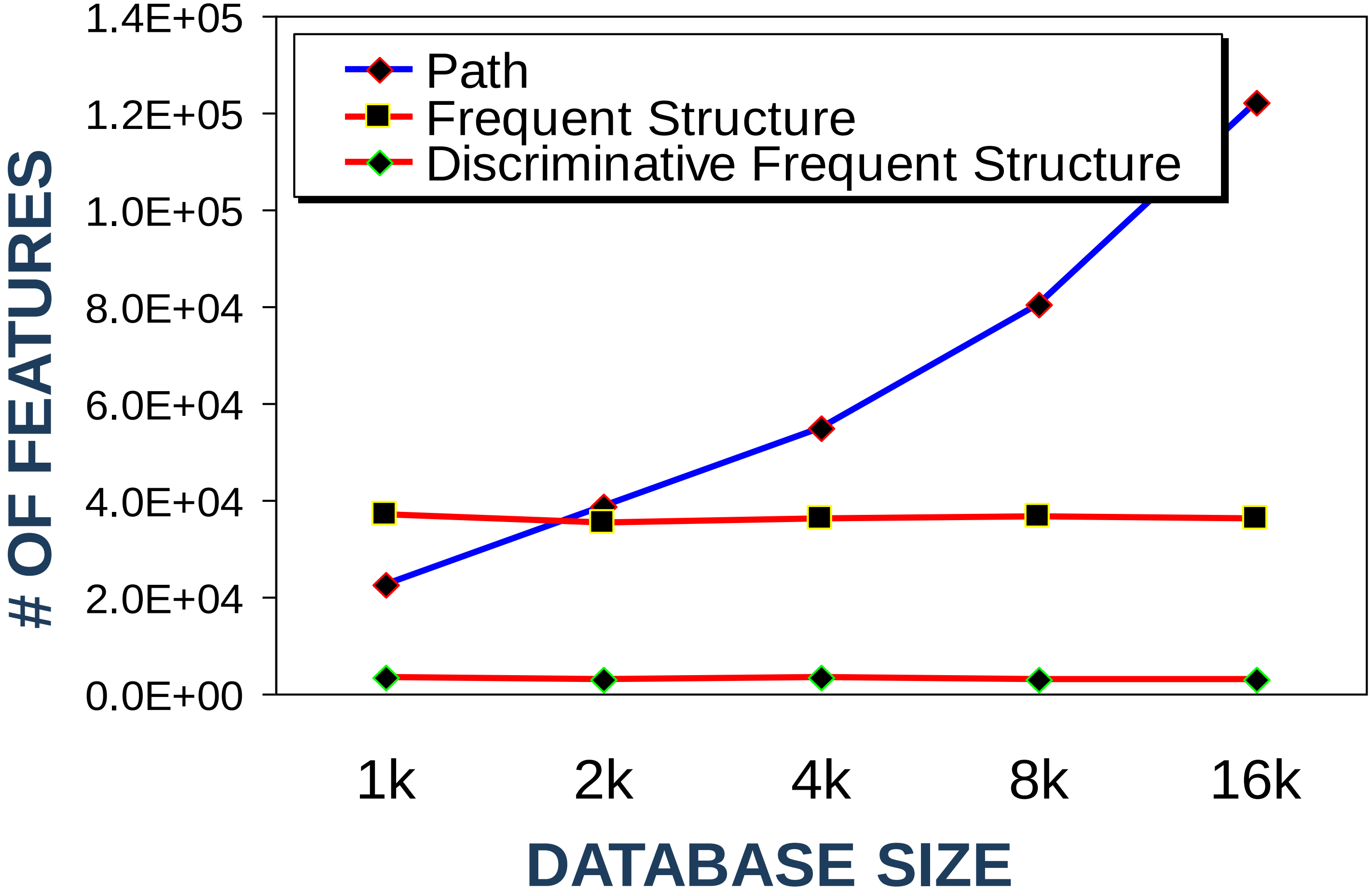
- We cannot index (or even search) all of substructures
- Large structures will likely be indexed well by their substructures
- Size-increasing support threshold



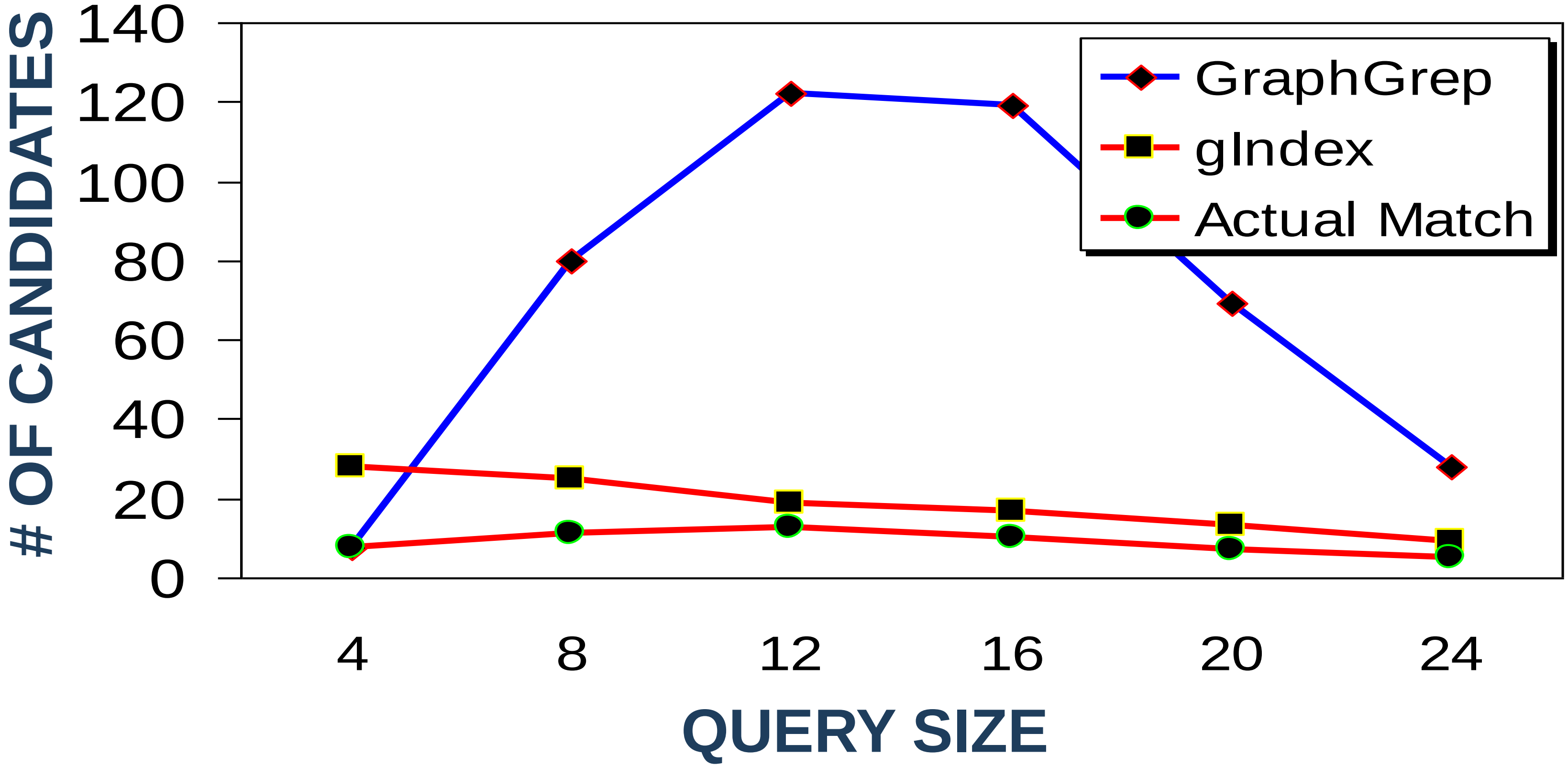
Experimental Setting

- The AIDS antiviral screen compound dataset from NCI/NIH, containing 43,905 chemical compounds
- Query graphs are randomly extracted from the dataset
- GraphGrep: maximum length (edges) of paths is set at 10
- gIndex: maximum size (edges) of structures is set at 10

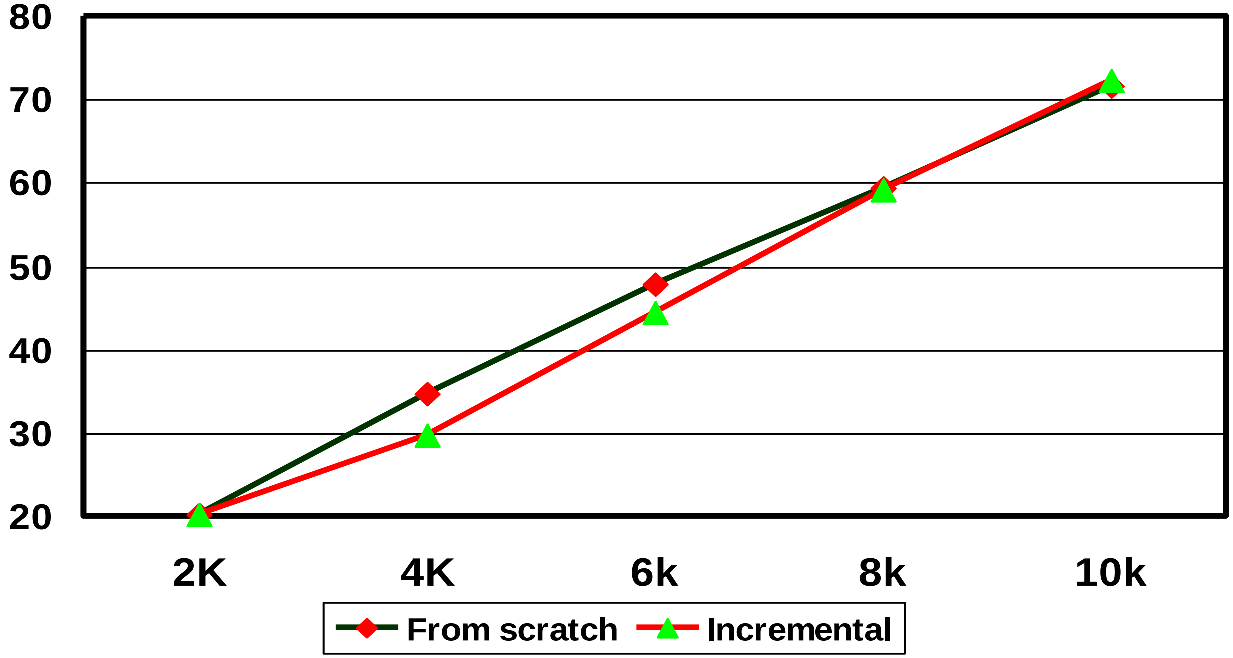
Experiments: Index Size



Experiments: Answer Set Size



Experiments: Incremental Maintenance



Frequent structures are stable to database updating
Index can be built based on a small portion of a graph database, but be used for the whole database

Graph Mining

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

Summary: Graph Mining

- 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 discriminative subgraphs are high-quality indexing features
- Similarity search in graph databases
  - Indexing and feature-based matching
- Further development and application exploration

References (1)

- T. Asai, et al. “Efficient substructure discovery from large semi-structured data”, SDM'02
- 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

References (3)

- 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
- M. Kuramochi and G. Karypis. "Frequent subgraph discovery", ICDM'01
- M. Kuramochi and G. Karypis, "GREW: A Scalable Frequent Subgraph Discovery Algorithm", ICDM'04
- C. Liu, X. Yan, H. Yu, J. Han, and P. S. Yu, "Mining Behavior Graphs for 'Backtrace" of Noncrashing Bugs'", SDM'05
- 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

References (4)

- 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
- 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. J. Zaki. “Efficiently mining frequent trees in a forest”, KDD'02