Lecture 8. Graph Pattern Mining

Lecture 8. Graph Pattern Mining

- Graph Pattern and Graph Pattern Mining
- Apriori-Based Graph Pattern Mining Methods
- gSpan: A Pattern-Growth-Based Method
- CloseGraph: Mining Closed Graph Patterns
- Graph Pattern Mining Application: Graph Indexing
- Mining Top-K Large Structural Patterns in a Massive Network

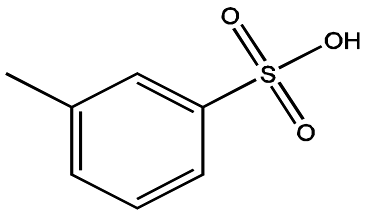
Thanks to Xifeng Yan@UCSB and Feida Zhu@SMU.SG for their contributions

Session 1. Graph Pattern and Graph Pattern Mining

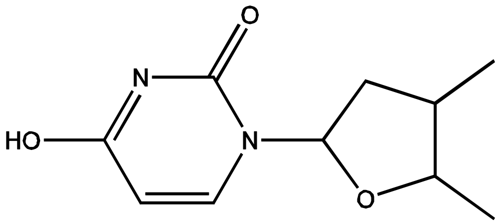
Frequent (Sub)Graph Patterns

- Given a labeled graph dataset D = {G_1, G_2, …, G_n), the supporting graph set of a subgraph g is $D_g = \{G_i \mid g \subseteq G_i, G_i \in D\}$.
  - support(g) = $|D_g| / |D|$
- A (sub)graph g is ***frequent*** if *support*(g) ≥ *min_sup*
- Ex.: Chemical structures
- Alternative:
  - Mining frequent subgraph patterns from a single large graph or network

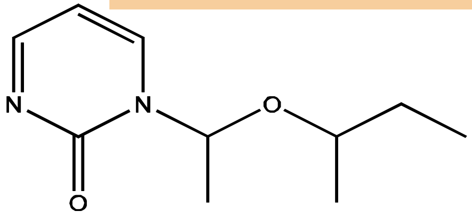
Graph Dataset



(A)



(B)

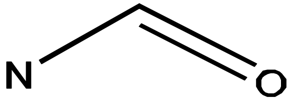


(C)

min_sup = 2

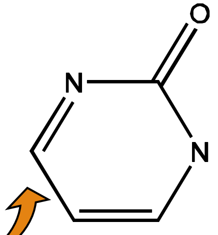
Frequent Graph Patterns

(1)



support = 67%

(2)



Applications of Graph Pattern Mining

- Bioinformatics
  - Gene networks, protein interactions, metabolic pathways
- Chem-informatics: Mining chemical compound structures
- Social networks, web communities, tweets, …
- Cell phone networks, computer networks, …
- Web graphs, XML structures, semantic Web, information networks
- Software engineering: program execution flow analysis
- Building blocks for graph classification, clustering, compression, comparison, and correlation analysis
- Graph indexing and graph similarity search

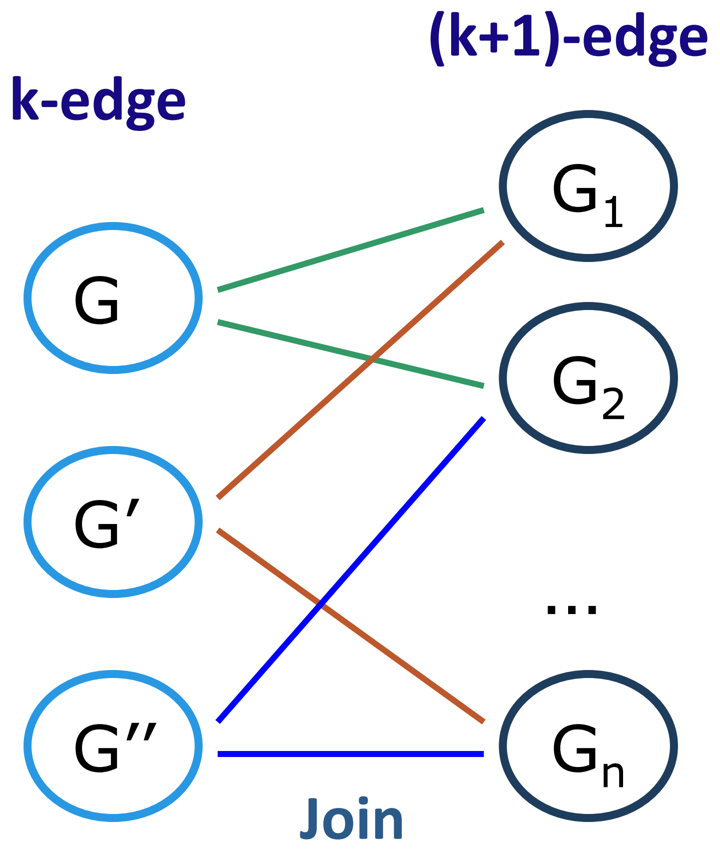
Graph Pattern Mining Algorithms: Different Methodologies

- Generation of candidate subgraphs
  - Apriori vs. pattern growth (e.g., FSG vs. gSpan)
- Search order
  - Breadth vs. depth
- Elimination of duplicate subgraphs
  - Passive vs. active (e.g., gSpan (Yan&Han'02))
- Support calculation
  - Store embeddings (e.g., GASTON (Nijssen&Kok'04, FFSM (Huan, et al.'03), MoFa (Borgelt and Berthold ICDM'02))
- Order of pattern discovery
  - Path → tree → graph (e.g., GASTON (Nijssen&Kok'04)

Session 2. Graph Pattern Mining: Apriori-Based Approach

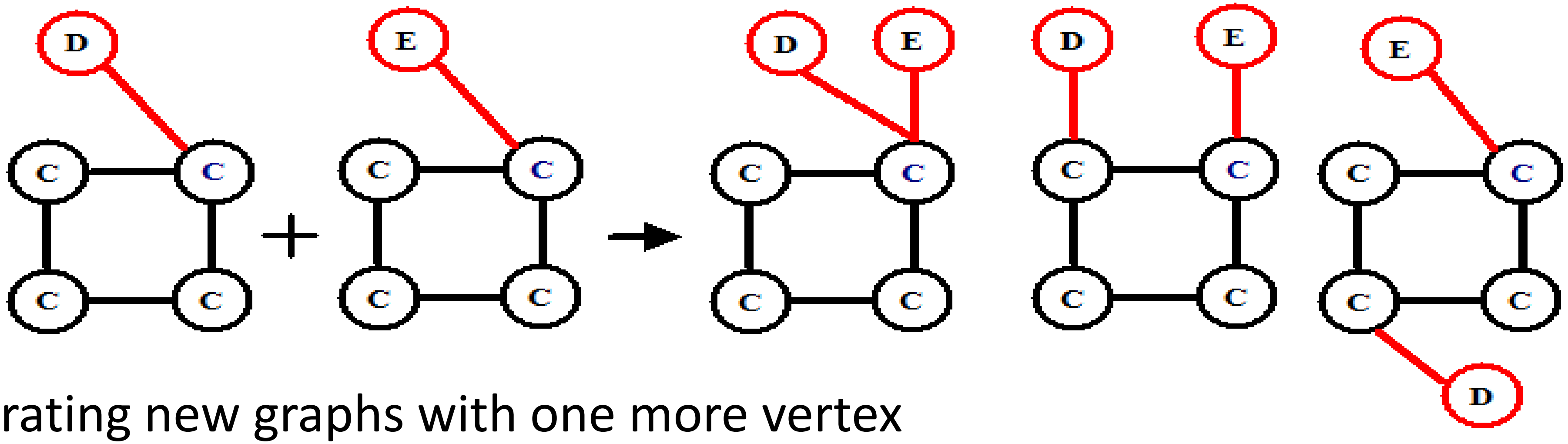
Apriori-Based Approach

- The Apriori property (anti-monotonicity): A size-*k* subgraph is frequent if and only if all of its subgraphs are frequent
- A candidate size-(*k*+1) edge/vertex subgraph is generated if its corresponding two *k*-edge/vertex subgraphs are frequent
- Iterative mining process:
  - Candidate-generation → candidate pruning → support counting → candidate elimination



Candidate Generation: Vertex Growing vs. Edge Growing

- Methodology: breadth-search, Apriori joining two size-*k* graphs
  - Many possibilities at generating size-(*k*+1) candidate graphs

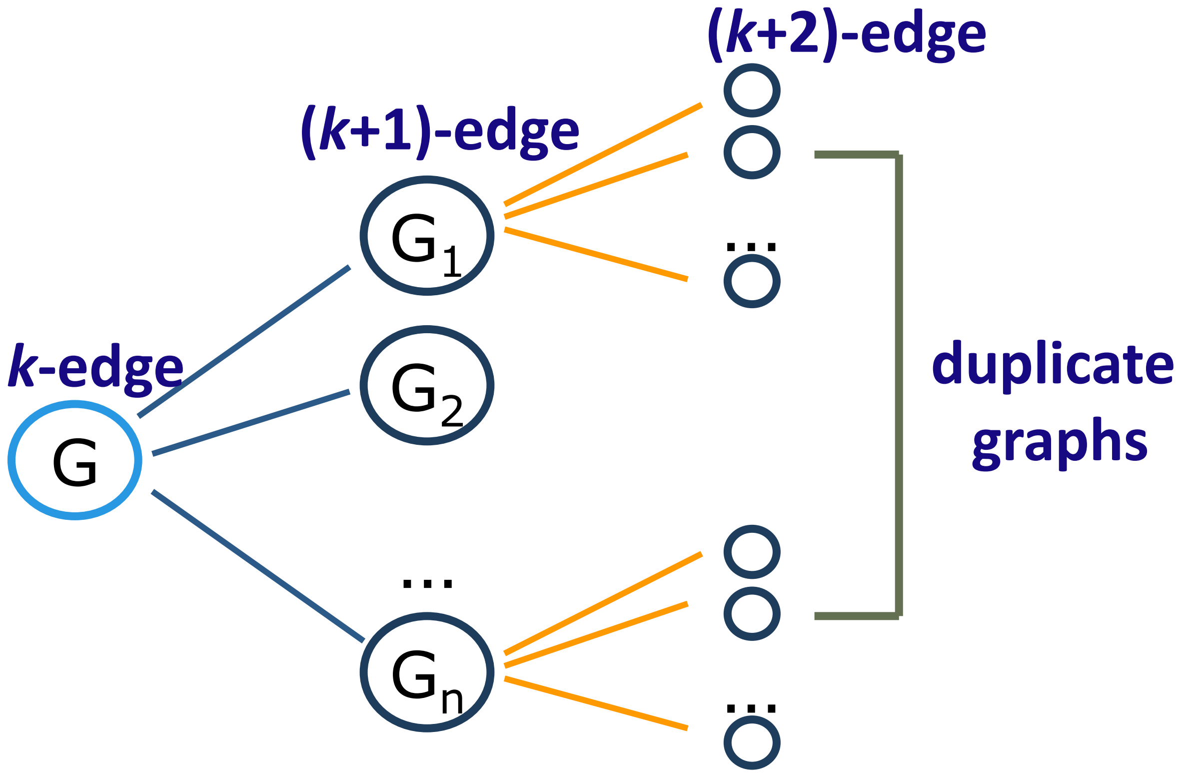


- Generating new graphs with one more vertex
  - AGM (Inokuchi, et al., PKDD'00)
- Generating new graphs with one more edge
  - FSG (Kuramochi and Karypis, ICDM'01)
- Performance shows *via edge growing* is more efficient

Session 3. gSpan: A Pattern Growth Approach

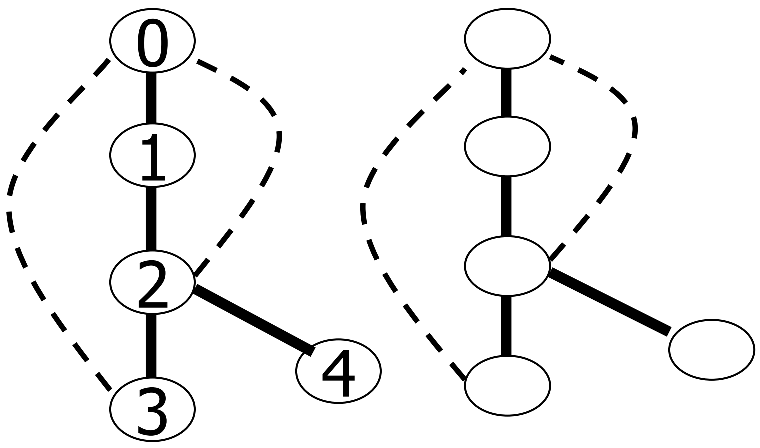
Pattern-Growth Approach

- Depth-first growth of subgraphs from *k*-edge to (*k*+1)-edge, then (*k*+2)-edge subgraphs
- Major challenge
  - Generating many duplicate subgraphs
- Major idea to solve the problem
  - Define an order to generate subgraphs
  - DFS spanning tree: Flatten a graph into a sequence using depth-first search
  - gSpan (Yan & Han: ICDM'02)



gSPAN: Graph Pattern Growth in Order

- **Right-most path extension** in subgraph pattern growth
  - Right-most path: The path from root to the right-most leaf (choose the vertex w. the smallest index at each step)
  - Reduce generation of duplicate subgraphs
- **Completeness:** The Enumeration of graphs using right-most path extension is <u>complete</u>
- DFS Code: Flatten a graph into a sequence using depth-first search

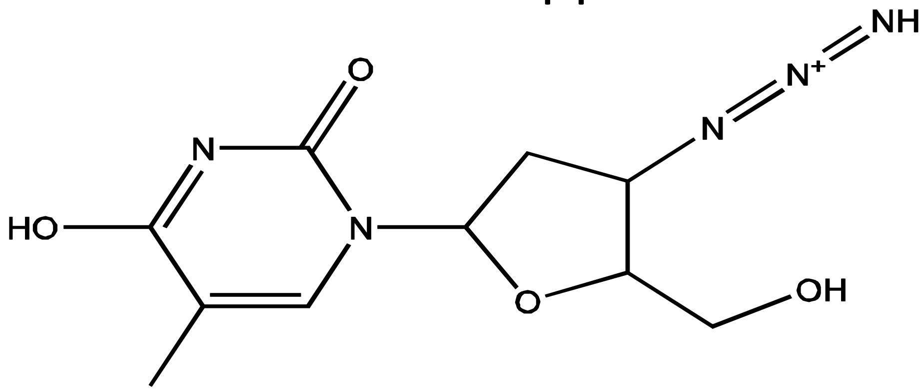


e_0: (0,1)
e_1: (1,2)
e_2: (2,0)
e_3: (2,3)
e_4: (3,0)
e_5: (2,4)

Session 4. CloseGraph: Mining Closed Graph Patterns

Why Mining Closed Graph Patterns?

- Challenge: An **n**-edge frequent graph may have 2^n subgraphs
- Motivation: Explore *closed frequent subgraphs* to handle graph pattern explosion problem
- A frequent graph G is *closed* if there exists no supergraph of G that carries the same support as G

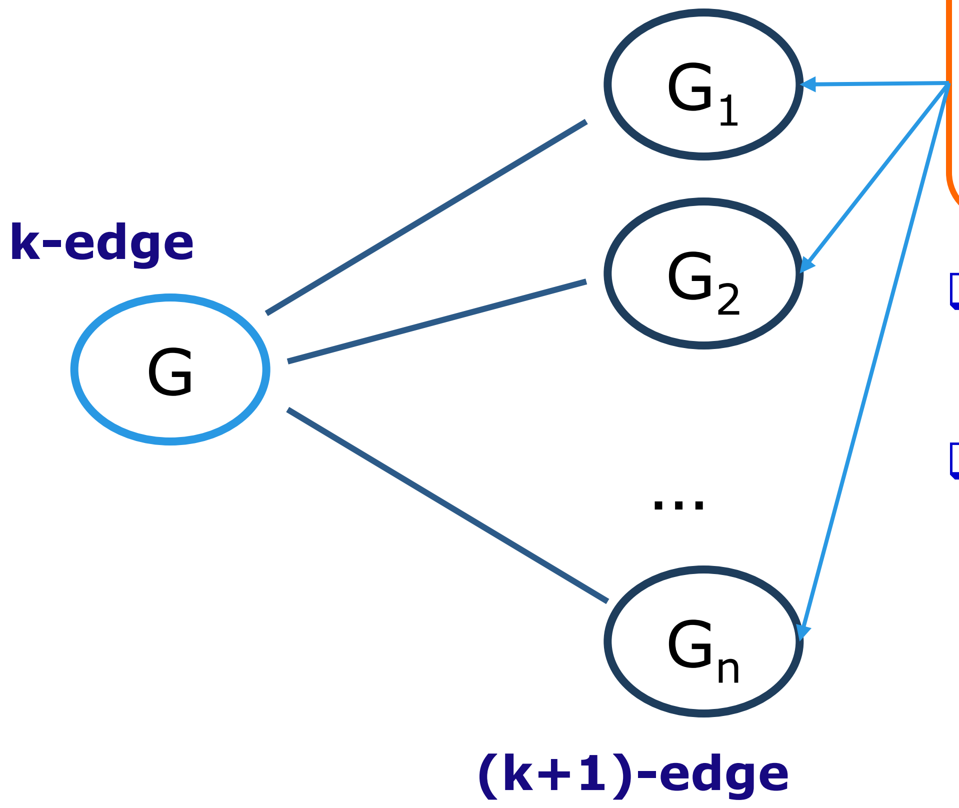


If this subgraph is *closed* in the graph dataset, it implies that none of its frequent super-graphs carries the same support

- *Lossless compression:* Does not contain non-closed graphs, but still ensures that the mining result is complete
- Algorithm CloseGraph: Mines closed graph patterns directly

CLOSEGRAPH: Directly Mining Closed Graph Patterns

- CloseGraph: Mining closed graph patterns by extending gSpan (Yan & Han, KDD'03)



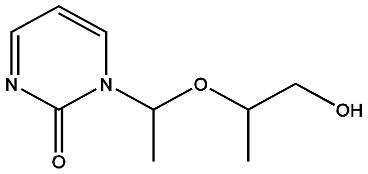
At what condition can we stop searching their children, i.e., early termination?

- Suppose G and G_1 are frequent, and G is a subgraph of G_1
- If **in any part of the graph in the dataset where G occurs, G_1 also occurs**, then we need not grow G (except some special, subtle cases), since none of G's children will be closed except those of G_1

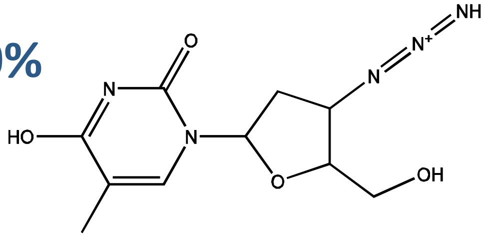
Experiment and Performance Comparison

- The AIDS antiviral screen compound dataset from NCI/NIH
- The dataset contains 43,905 chemical compounds
- Discovered Patterns: The smaller minimum support, the bigger and more interesting subgraph patterns discovered

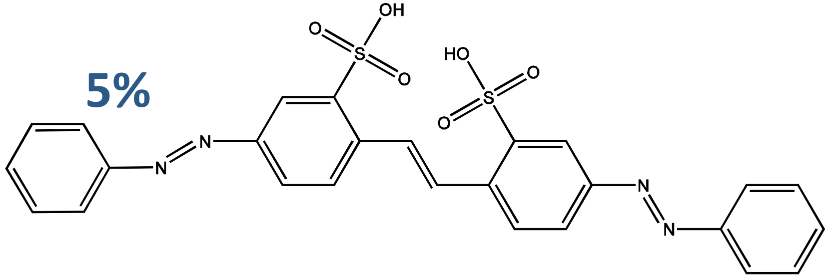
20%



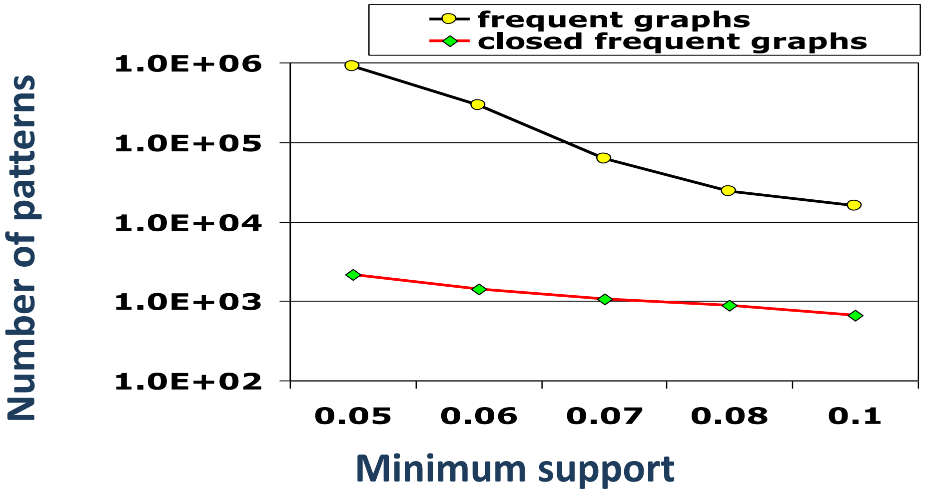
10%



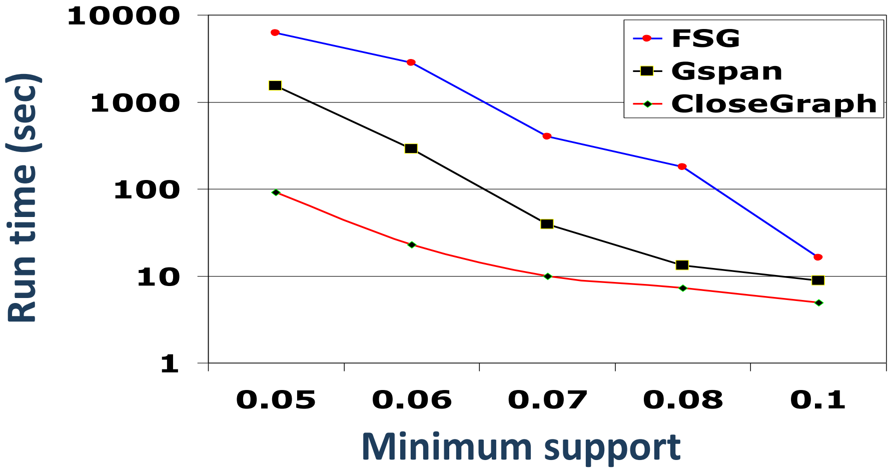
5%



# of Patterns: Frequent vs. Closed



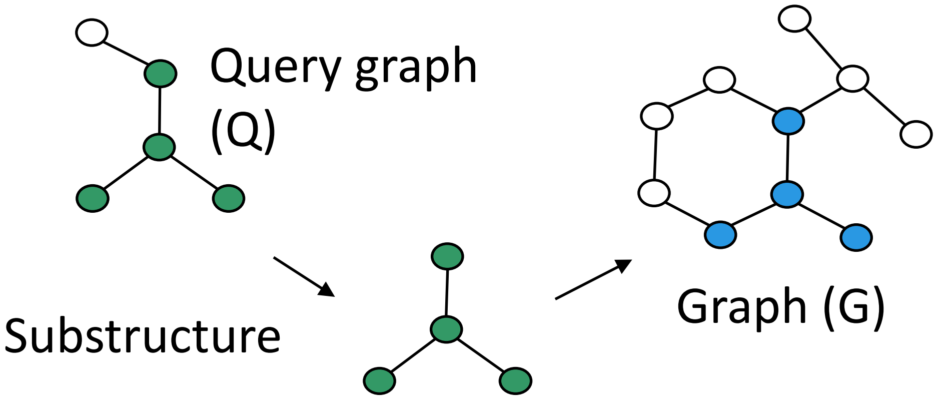
Runtime: Frequent vs. Closed



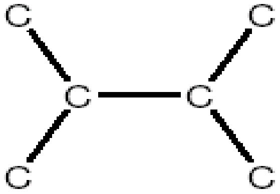
Session 5. gIndex: A Graph Indexing Method

Application of Pattern Mining: Graph Indexing

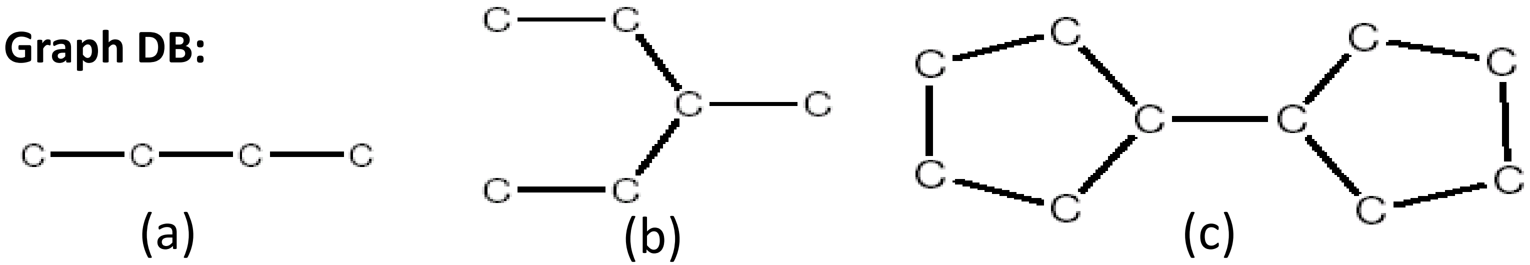
- Graph query: Find all the graphs in a graph DB containing a given query graph
- Index should be a powerful tool
- Path-index may not work well
- Solution: Index directly on substructures (i.e., graphs)



Query Q:



Graph DB:



Only graph (c) contains Q

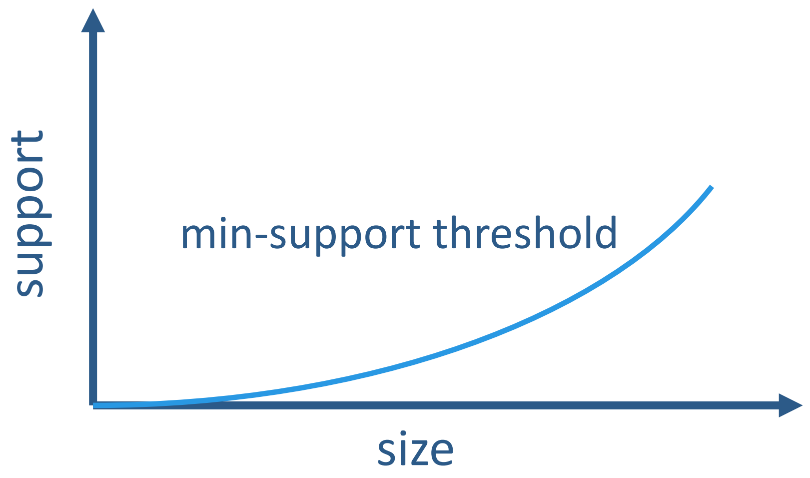
Path-indices: C, C-C, C-C-C, C-C-C-C cannot prune (a) & (b)

gIndex: Indexing Frequent and Discriminative Substructures

- Why index frequent substructures?
  - Too many substructures to index
  - Size-increasing support threshold
  - Large structures will likely be indexed well by their substructures
- Why discriminative substructures?
  - Reduce the index size by an order of magnitude
- Selection: Given a set of selected structures $f_1, f_2, .. f_n$, and a new structure *x*, the extra indexing power is measured by

$$\Pr\left(x \middle| f_1, f_2, \ldots f_n\right), f_i \subset x$$

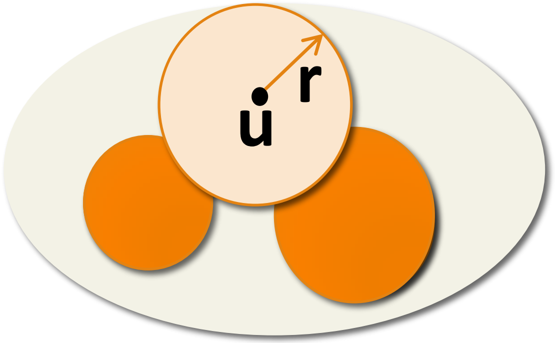
when $\Pr(x|f_1, f_2, \ldots, f_n)$ is small enough, *x* is a discriminative structure and should be included in the index
- Experiments show gIndex is small, effective and stable



Session 6. SpiderMine: Mining Top-K Large Structural Patterns in a Single Network

SpiderMine: Mining Top-K Large Structural Patterns in a Massive Network

- Large patterns are informative to characterize a large network (e.g., social network, web, or bio-network)
- Similar to pattern fusion, mining large pattern should not aim for completeness but for representativeness of the target results
- Spider-Mine (F. Zhu, et al., VLDB'11): Mine top-*K* largest frequent substructure patterns whose diameter is bounded by D_{max} with a probability at least $1-\epsilon$
- General idea: Large patterns are composed of a number of small components ("spiders") which will eventually connect together after some rounds of pattern growth
- **r-Spider:** An r-spider is a frequent graph pattern P such that there exists a vertex u of P, and all other vertices of P are within distance r from u

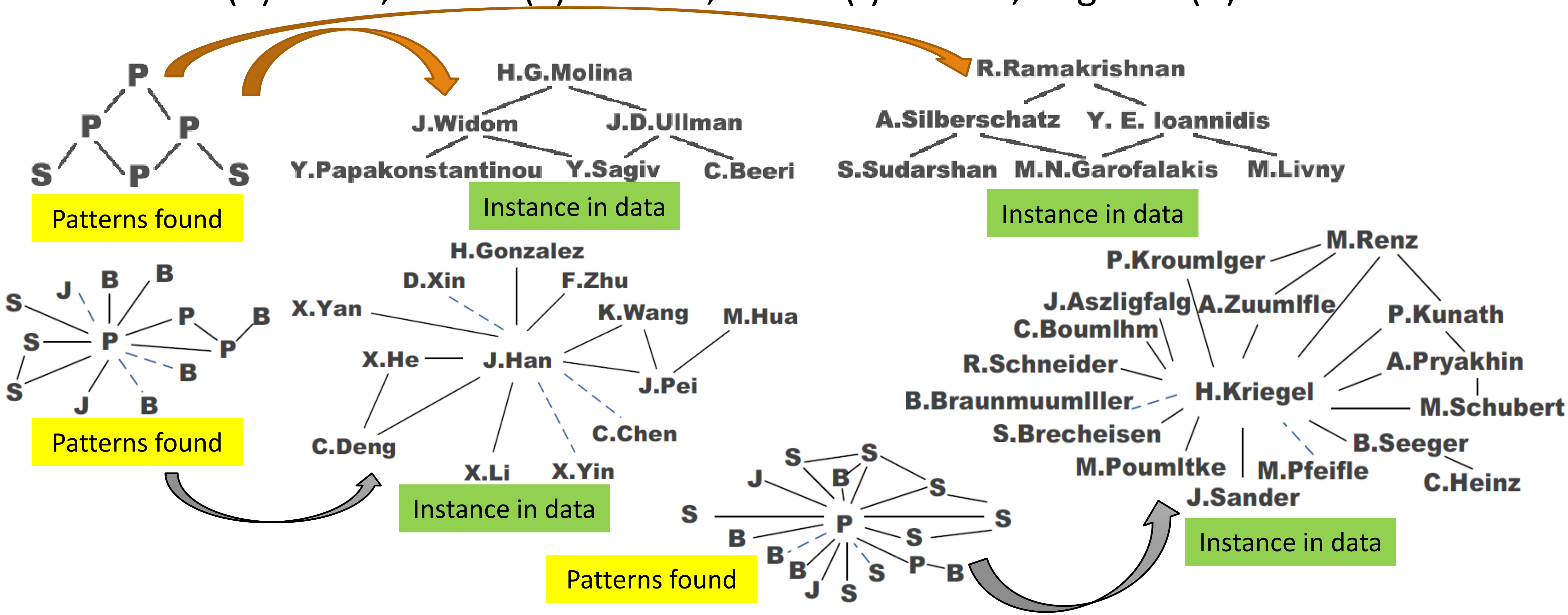


Why Is SpiderMine Good for Mining Large Patterns

- The SpiderMine Algorithm
  - Mine the set S of all the r-spiders
  - Randomly draw M r-spiders
  - Grow these M r-spiders for t = $D_{max}/2$ iterations, and merge two patterns whenever possible
  - Discard unmerged patterns
  - Continue to grow the remaining ones to maximum size
  - Return the top-K largest ones in the result
- Why is SpiderMine likely to retain large patterns and prune small ones?
  - Small patterns are much less likely to be hit in the random draw
  - Even if a small pattern is hit, it is even less likely to be hit multiple times
  - The larger the pattern, the greater the chance it is hit and saved

Mining Collaboration Patterns in DBLP Networks

- Data description: 600 top confs, 9 major CS areas, 15071 authors in DB/DM
- Author labeled by # of papers published in DB/DM
  - Prolific (P): >=50, Senior (S): 20~49, Junior (J): 10~19, Beginner(B): 5~9



Summary

- Graph pattern mining: Basic concepts
- Apriori-based graph pattern mining methods
- gSpan: A pattern-growth-based method
- CloseGraph: Mining closed graph patterns
- Graph Indexing: A graph pattern mining application example
- SpiderMine: Mining top-k large structural patterns in a large network

Recommended Readings

- C. Borgelt and M. R. Berthold, "Mining molecular fragments: Finding relevant substructures of molecules", ICDM'02
- J. Huan, W. Wang, and J. Prins. "Efficient mining of frequent subgraph in the presence of isomorphism", ICDM'03
- A. Inokuchi, T. Washio, and H. Motoda. "An apriori-based algorithm for mining frequent substructures from graph data", PKDD'00
- M. Kuramochi and G. Karypis. "Frequent subgraph discovery", ICDM'01
- S. Nijssen and J. Kok. A quickstart in frequent structure mining can make a difference. KDD'04
- N. Vanetik, E. Gudes, and S. E. Shimony. "Computing frequent graph patterns from semistructured data", ICDM'02
- 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
- F. Zhu, Q. Qu, D. Lo, X. Yan, J. Han, and P. S. Yu, "Mining Top-K Large Structural Patterns in a Massive Network", VLDB'11