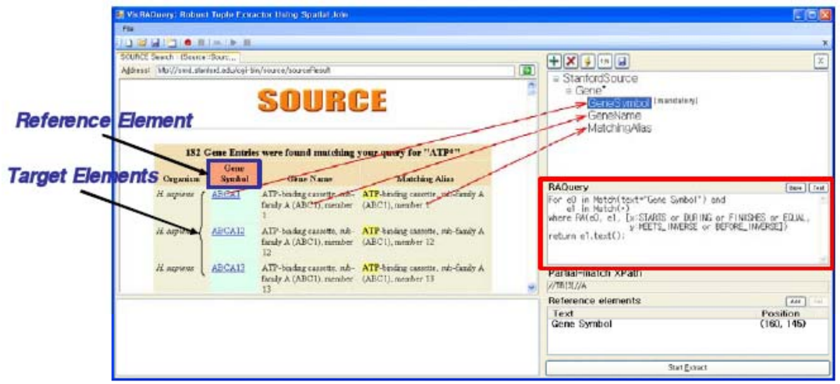
2009년 연구 결과

- 두 편의 데이터 베이스 분야의 Top Conference 논문 게재 및 게재 승인
  - Wook-Shin Han and Jinsoo Lee, "*Dependency-Aware Reordering for Parallelizing Query Optimization in Multi-Core CPUs*," In SIGMOD 2009.
  - Wook-Shin Han, Wooseong Kwak, and Hwanjo Yu, "*On Supporting Effective Web Extraction*," In ICDE 2010.



Graph Indexing: A Frequent Structure-based Approach

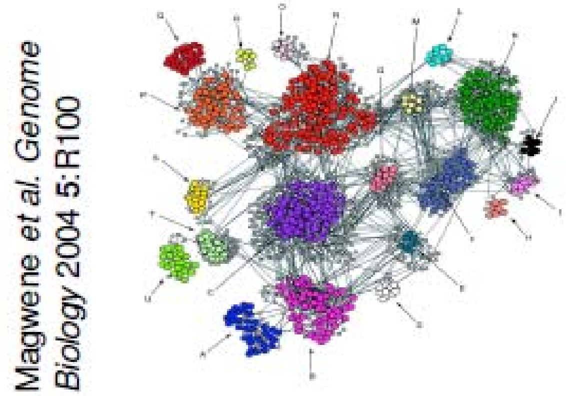
Presenter: Jinsoo Lee
Kyungpook National University
January 2010

Xifeng Yan, Philip S. Yu, and Jiawei Han, "Graph Indexing: A Frequent Structure-based Approach," In SIGMOD 2004

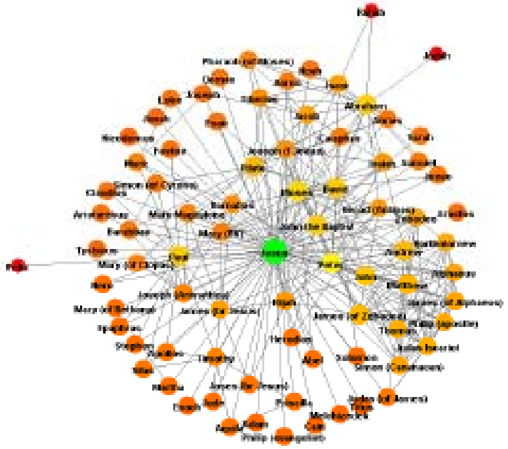
Why Graph Searching?

- Graphs are ubiquitous
  - Chemical compounds (Cheminformatics)
  - Protein structures, biological pathways/networks (Bioinformatics)
  - Program control flow, traffic flow, and work flow analysis
  - XML databases, Web, and social network analysis
- Graph is a general model
  - Trees, lattices, sequences, and items are degenerated graphs
- Complexity of algorithms: many problems are of high complexity

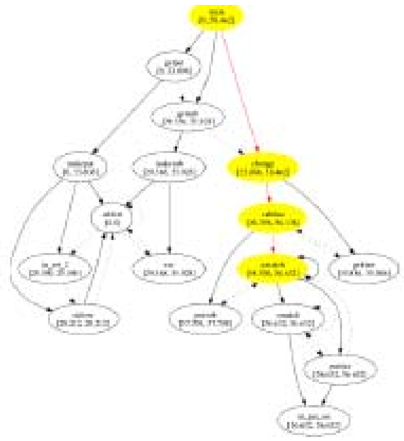
Graphs are Everywhere



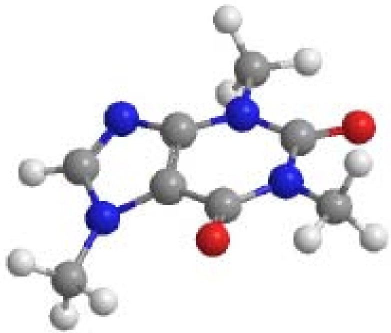
Co-expression Network



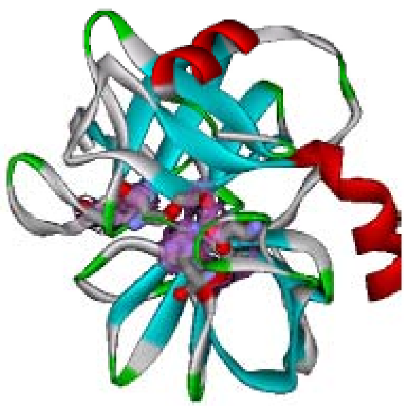
Social Network



Program Flow



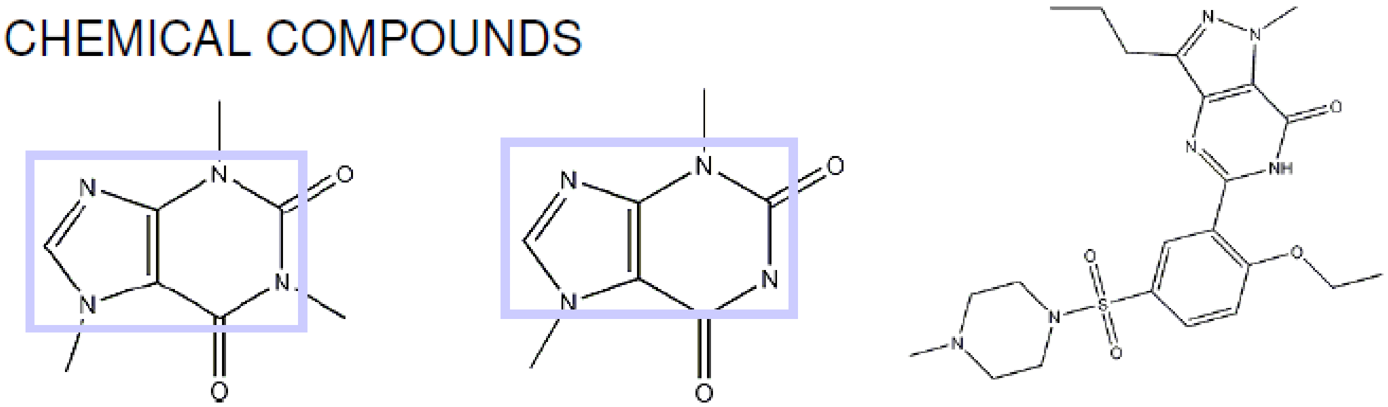
Chemical Compound



Protein Structure

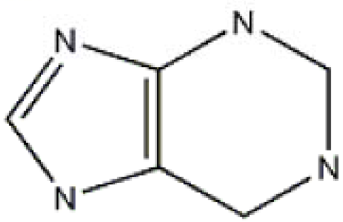
Example 1: Frequent Subgraphs

CHEMICAL COMPOUNDS



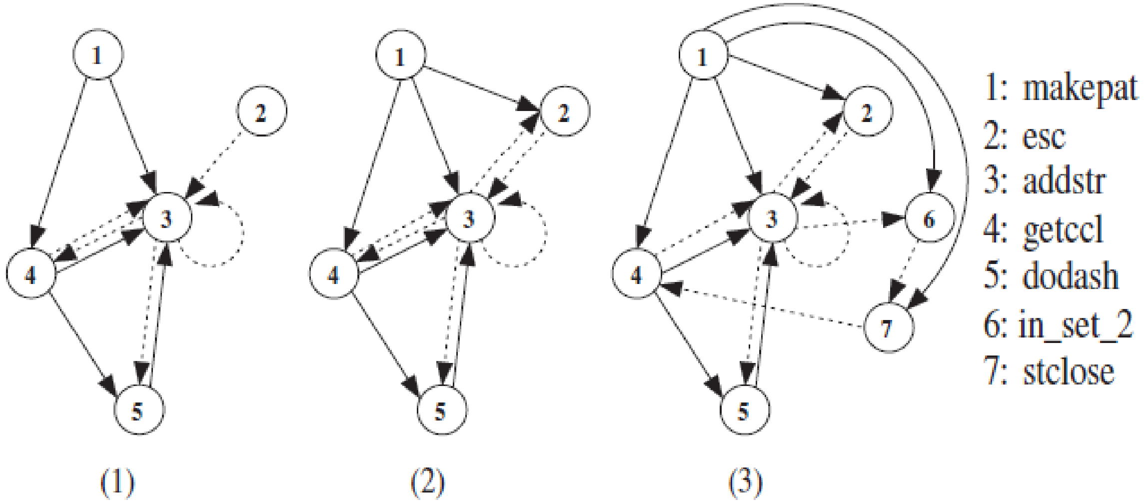
(a) caffeine (b) diurobromine (c) viagra ...

FREQUENT SUBGRAPH

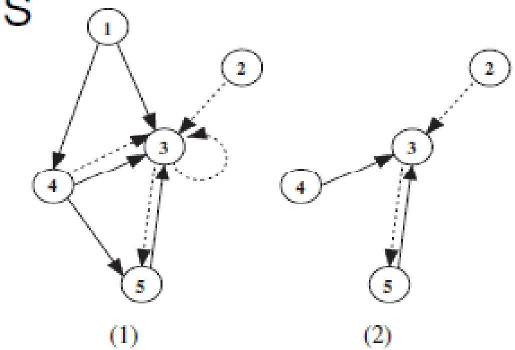


Example 2: Frequent Subgraphs

PROGRAM CALL GRAPHS

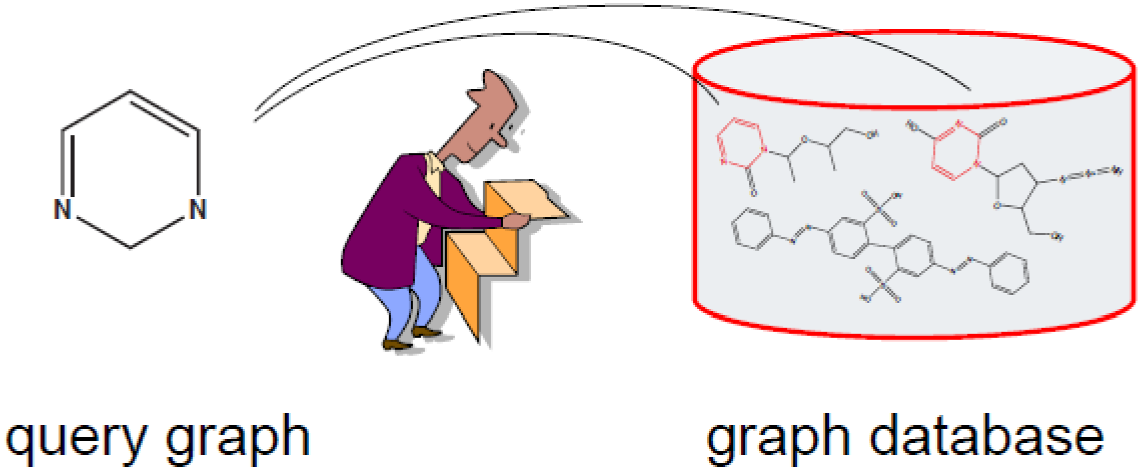


FREQUENT SUBGRAPHS
(MIN SUPPORT IS 2)



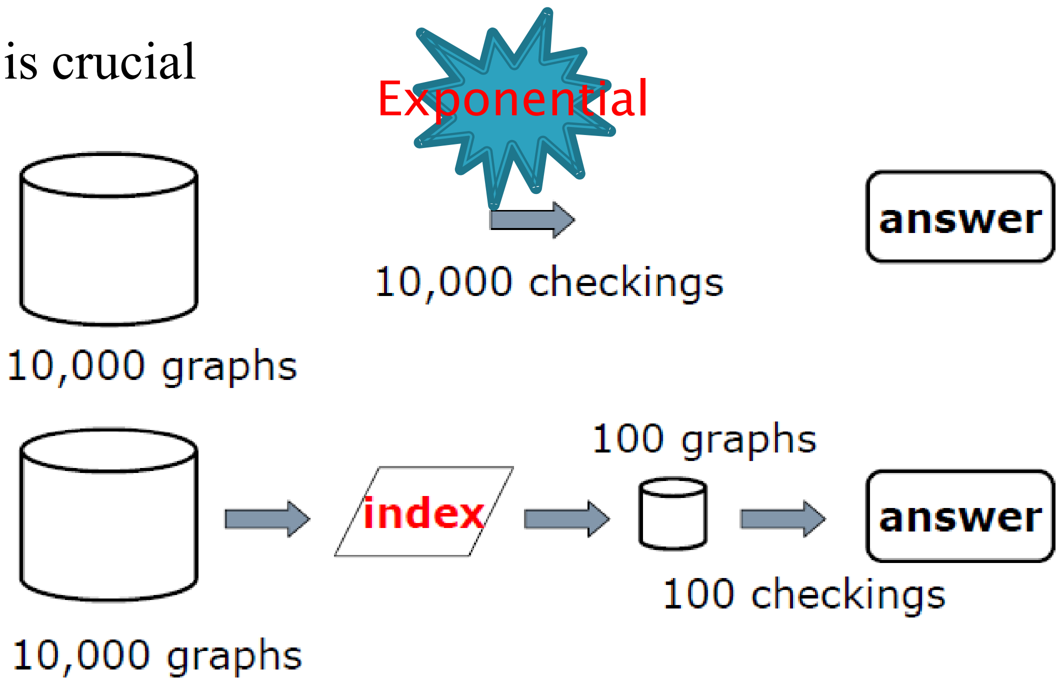
Graph Searching

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



Indexing Graphs

- Indexing is crucial



- Many graph indexing techniques are introduces.
  - GraphGrep, ***gIndex***, Closure-Tree, Tree+Delta, GCoding, FG-Index, TreePI, SWIFT-Index …

Problem Definition

- Find all of the graphs in a database that contain the query graph
- Given a graph database $D = \{g_1, g_2, ..., g_n\}$ and a query graph q, finds the query answer set $D_q=\{g_i|q\subseteq g_i, g_i\in D\}$
- (Example) For the query q shown in Figure 2, the query answer set, D_q, has only one element: graph (c) in Figure 1.

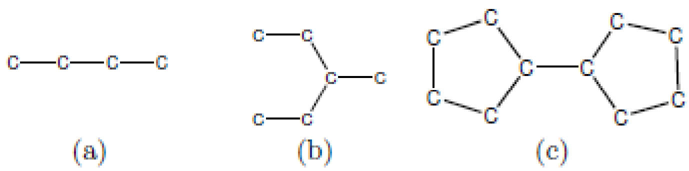


Figure 1: A Sample Database

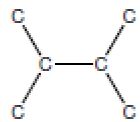


Figure 2: A Sample Query

Overview of the Framwork

- Index construction (preprocessing step)
  - Enumerating all sub-graphs in a database D.
  - Selecting *graph feature set F* from the results of the first step.
    - The gIndex selects graph features which are frequent and discriminative
  - Indexing graph feature set F to find $D_f = \{g_i | f \subseteq g_i, g_i \in D\}$ efficiently for given $f \in F$.
    - Indexing selected graph features as a prefix tree

- Query processing
  - Filter: enumerates all the features in a query graph q, and finds *candidate query answer set* $C_q = \cap_f D_f$ $(f \subseteq q \wedge f \in F)$ using the index.
  - Verify: verify graph g in C_q whether q is really a sub-graph of g (sub-graph isomorphic test).

Frequent Fragment

- *Support*(g): the number of graphs in D where g is a sub-graph ($=|D_g|$).

- Frequent graph (Fragment): if *support*(g) $\geq$ *minSup*, graph g is frequent.
  - (Example) graphs (a) and (b) in Figure 3 are frequent, if *minSup* = 2.

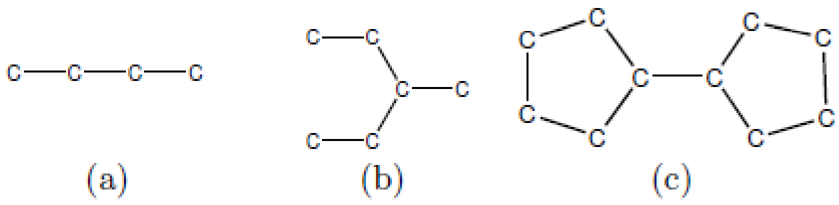


Figure 1: A Sample Database

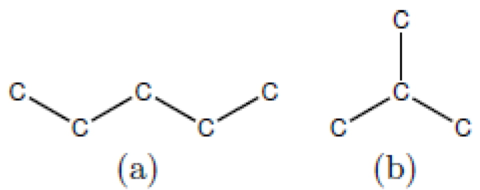
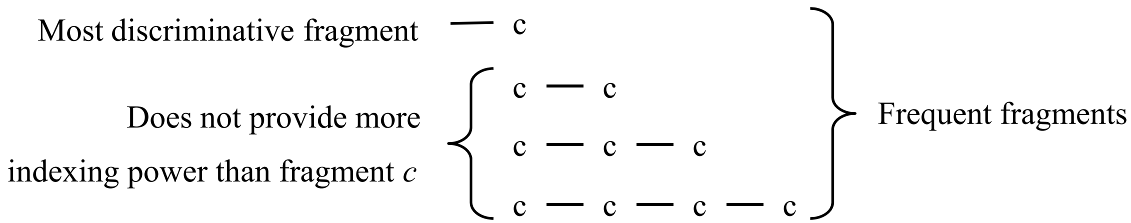


Figure 3: Frequent Fragments

Discriminativity

- $f \subseteq f'$ and $D_f = D_f$ (and *support*(f) = *support*(f'))
  - f' does not provide more information than f if both are selected as indexing features.
  - f is more *discriminative* than f.
  - f' should be removed from the feature set.



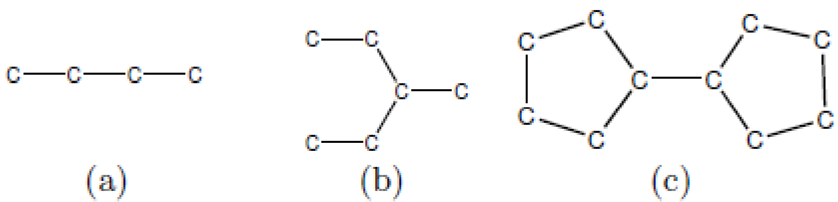


Figure 1: A Sample Database

Discriminative Ratio

- The measurement of the discriminativeness of a fragment *x*.
- Calculated by the following formula:

A set of graphs which contain the sub-graphs of *x* in the feature set

$$\gamma = \frac{\left|\bigcap_i D_{f_{\varphi_i}}\right|}{\left|D_x\right|}, f_{\varphi_i} \subseteq x, f_{\varphi_i} \in F, 1 \leq i \leq n$$

A set of graphs containing graph *x*

- Properties:
  - $\gamma \geq 1$.
  - when $\gamma = 1$, fragment *x* is completely redundant.
  - when $\gamma >> 1$, fragment *x* is more discriminative than the combination of fragments $f_{\varphi i}$.
- **Select frequent fragments whose γ is no less than pre-defined minimum discriminative ratio γ_{min}(=2.0).**

Index Construction

- Translates fragments into sequences and holds them in a prefix tree.
- Each fragment is associated with (graph) id list (ids of graphs containing this fragment).

- Translating fragments: use the canonical DFS coding in *gSpan* [Yan02].
- Prefix tree: gIndex Tree proposed in this paper.

Algorithm – Search

Algorithm 2 Search

Input: Graph database D, Feature set F, Query q, and Maximum fragment size $maxL$.
Output: Candidate answer set C_q.

```
let C_q = D;
for each fragment x ⊆ q and len(x) ≤ maxL do
    if x ∈ F then
        C_q = C_q ∩ D_x;
return C_q;
```

- Input: graph database D, feature set F, query q, and maximum fragment size $maxL$.
- Output: candidate answer set C_q.

- Algorithm
  1. Enumerates all fragments up to a maximum size.
  2. Find graph id list using the gIndex.
  3. Intersects the id lists associated with these fragments.

- Optimization
  - Apriori pruning
  - Maximum discriminative fragments
  - Inner support

References

- [Sha02] D. Shasha, J. T-L Wnag, and R. Giugno. Algorithmics and applications of tree and graph searching, In PODS 2002.
- [Yan02] X. Yan and J. Han, gSpan: Graph-based substructure pattern mining, In ICDM 2002.