Querying Large Graph Databases

Yiping Ke
Chinese Univ. of Hong Kong
ypke@se.cuhk.edu.hk

James Cheng
Nanyang Technological Univ.
jamescheng@ntu.edu.sg

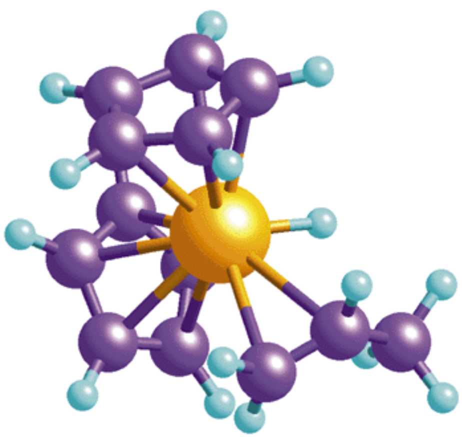
Jeffrey Xu Yu
Chinese Univ. of Hong Kong
yu@se.cuhk.edu.hk

香港中文大學
The Chinese University of Hong Kong

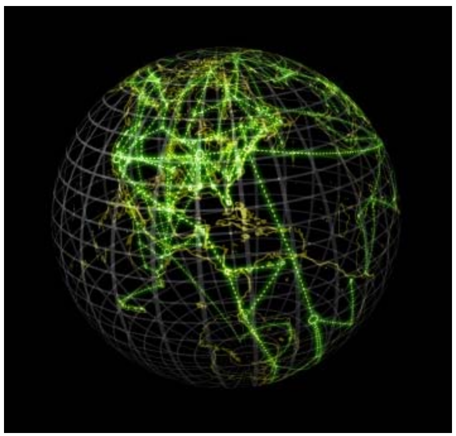
NANYANG TECHNOLOGICAL UNIVERSITY

Graph Data

- Graph is a powerful modeling tool
- Graph data is everywhere (e.g. chemistry, biology, image, vision, social networks, the Web, etc.)



Chemical bonds



Internet

DNA

Daily-life objects

Graph Data

- Volume of graph data grows rapidly in recent years
- SCI Finder report: 4000 new compound structures are added each day
- Demand for more efficient techniques for querying large graph databases

Graph Queries

- Graph queries in real applications
  - Chemical informatics and bio-informatics:
    - Graphs model compounds and proteins
    - Graph queries can be used for screening, drug design, motif discovery in 3D protein structures, protein interaction analysis, etc.
  - Computer vision:
    - Graphs represent organization of entities in images
    - Graph queries can be used to identify objects and scenes

Graph Queries

- Graph queries in real applications
  - Heterogeneous web-based data sources and e-commerce sites:
    - Graphs model schemas
    - Graph matching solves the problem of schema matching and integration
  - Others: program flows, software and data engineering, taxonomies, etc

Tutorial Coverage

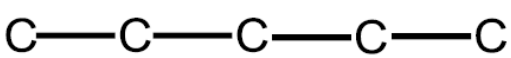
- Transaction graph databases
  - Containing a set of relatively small graphs
  - Mostly in scientific domains, e.g., chemistry and bioinformatics
  - Query types:
    - Subgraph queries
    - Supergraph queries
    - Similarity queries
- Other graph data such as large networks, see [Faloutsos and Tong, ICDE'09]

Tutorial Coverage

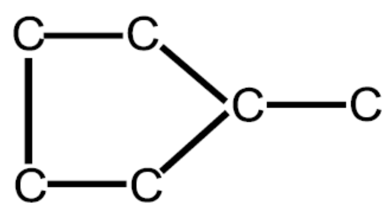
- Subgraph queries
- Supergraph queries
- Similarity queries

Subgraph Query Processing

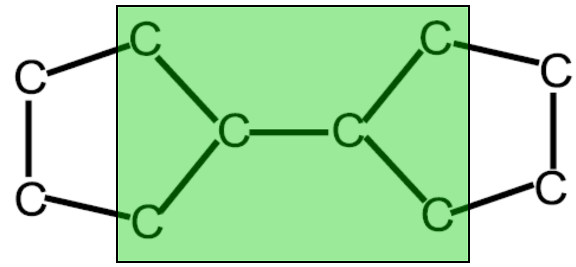
- Problem definition
  - Given a graph database D and a graph query q
  - Find all graphs g in D s.t. q is a subgraph of g



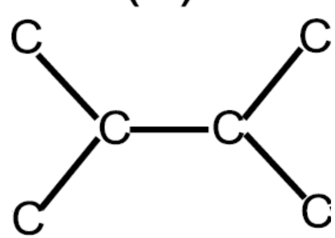
(a)



(b)



(c)



q

Applications

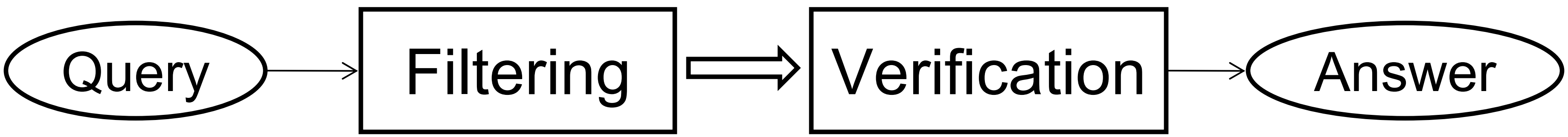
- Protein interaction analysis
- Motif discovery in 3D protein structures
- Drug design
- Schema matching
- Graph similarity search
- Correlation discovery in graph databases

Challenges

- Sub-problem: subgraph isomorphism (sub-Iso) => NP-complete
- Sequential scan of D + pair-wise comparison between q and each g in D
  => |D| sub-Iso tests
- Each g in D is relatively small but
  inefficient for large D or online applications

Existing Solution

- Filtering and Verification



- Filtering: filter false answers by an index and produce a candidate set C
- Verification: verify if $q \subseteq g$, for each $g \in C$ (by sub-Iso test)

Query Processing Cost

- Let C be the candidate set obtained by filtering using an index

  Cost = Cost(index-probing) +
  Cost(disk I/O) x |C| +
  Cost(sub-Iso) x |C|

- Objectives of existing indexes:
  - Keep a low Cost(index-probing)
  - Minimize |C|

Representative Work

- Feature-based approach
- Closure-based approach
- Verification-free approach
- Coding-based approach
- Fast sub-Iso approach

Representative Work

- Feature-based approach:
  - Select a set of features, F
  - Filtering by inclusion logic: for each $g \in D$, if $\exists f \in F$ such that $f \subseteq q$ and $f \nsubseteq g$, then $q \nsubseteq g$ and we filter out g
- Closure-based approach:
  - Index database based on graph closure
- Verification-free approach:
  - Attempt to totally eliminate the candidate set => no verification
- Coding-based approach:
  - Encode the graphs/query for more efficient matching
- Fast sub-Iso approach:
  - Speed up sub-Iso in the verification/filtering steps

	Feature-based	Closure-based	Verification-free	Coding-based	Fast sub-Iso
GraphGrep [Shasha et al., PODS'02]	X				
gIndex [Yan et al., SIGMOD'04]	X				
C-tree [He and Singh, ICDE'06]		X			X
FG-index [Cheng et al., SIGMOD'07]	X	X	X		
GString [Jiang et al., ICDE'07]				X	
TreePi [Zhang et al., ICDE'07]	X				X
GDIndex [Williams et al., ICDE'07]			X		
Tree+Δ [Zhao et al., VLDB'07]	X				
GCoding [Zou et al., EDBT'08]				X	
QuickSI [Shang et al., VLDB'08]	X				X

Representative Work

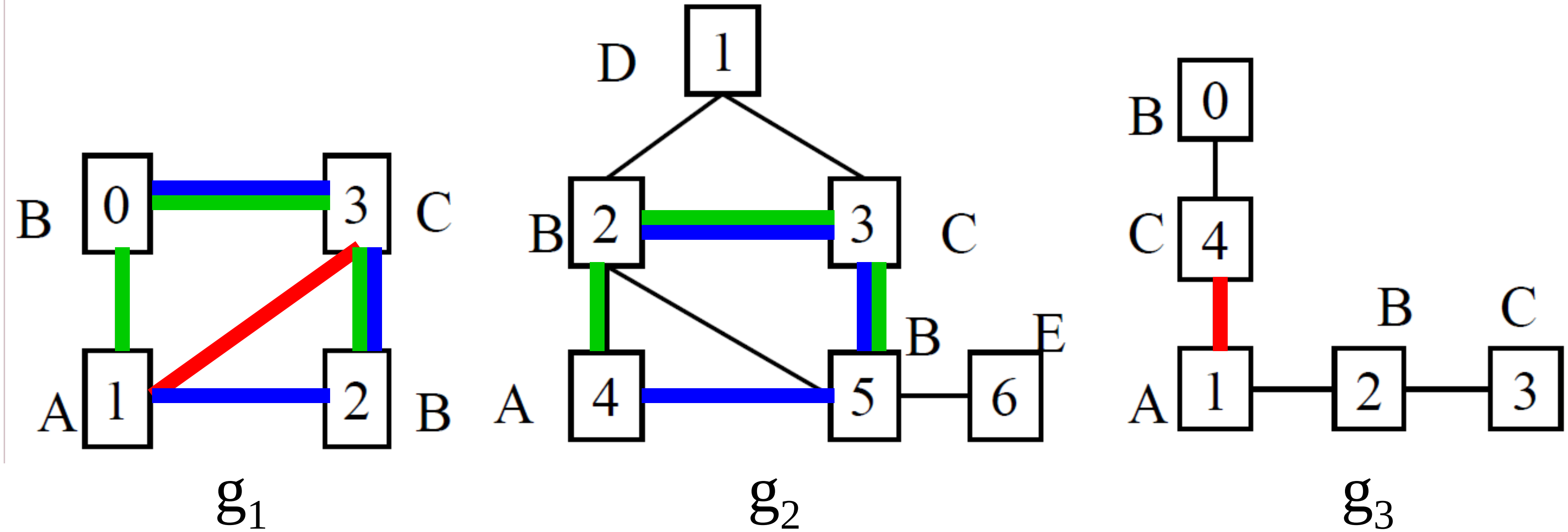
- Feature-based approach
  - GraphGrep [Shasha et al., PODS'02]
  - gIndex [Yan et al., SIGMOD'04]
  - TreePi [Zhang et al., ICDE'07]
  - Tree+Δ [Zhao et al., VLDB'07]
  - Others: FG-index, QuickSI
- Closure-based approach
- Verification-free approach
- Coding-based approach
- Fast sub-Iso approach

GraphGrep [Shasha et al., PODS'02]

- First work adopts the filtering-and-verification framework for subgraph query processing
- Motivation: sequential scan too expensive => reduce candidate set size by filtering
- Main idea: filtering by paths

GraphGrep [Shasha et al., PODS'02]

- Index construction
  - Enumerate the set of all paths, of length up to L, of all graphs in the database
  - Keep these paths in a hashtable

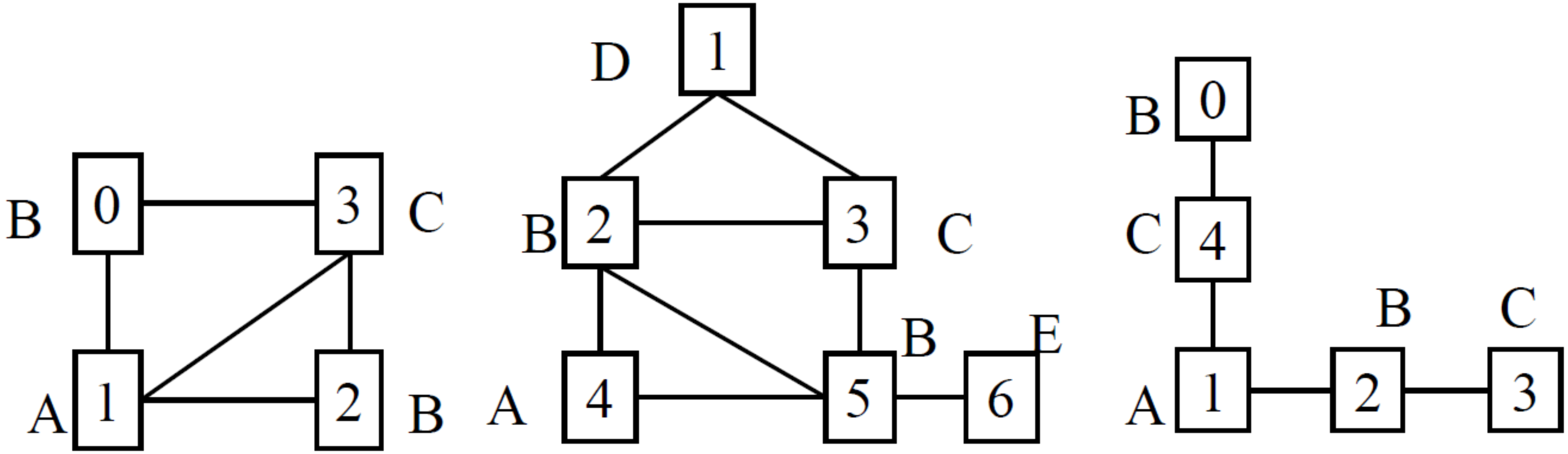


Key	g_1	g_2	g_3
h(CA)	1	0	1
……			
h(ABCB)	2	2	0

Index
(hashtable of paths)

GraphGrep [Shasha et al., PODS'02]

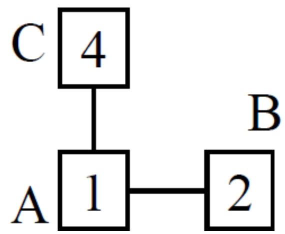
- Query processing
  - Filtering:
    - Hash all paths, of length up to L, of a query q
    - Filter out graphs that do not contain all paths in q
    - Filter by inclusion logic:
      - F: the set of features, i.e., paths
      - D_f: projected database of f, i.e., the set of graphs in D that are supergraphs of f
      - $C = \cap_{f \subseteq q \wedge f \in F} D_f$
  - Verification:
    - Test sub-Iso between q and each $g \in C$



g_1 g_2 g_3

Key	g_1	g_2	g_3
h(CA)	1	0	1
……			
h(ABCB)	2	2	0

Index
(hashtable of paths)



q

Filtering:

- $D_{CA}=\{g_1, g_3\}$
- $D_{BA}=\{g_1, g_2, g_3\}$
- $D_{CAB}=\{g_1, g_3\}$
- $C=D_{CA} \cap D_{BA} \cap D_{CAB}$
  $=\{g_1, g_3\}$

Verification:

- Do sub-Iso for (q, g_1) and (q, g_3)

Answer: $\{g_1, g_3\}$

GraphGrep [Shasha et al., PODS'02]

- Strengths
  - Indexing paths with length limit is fast
  - Index size is small
- Limitations
  - Filtering power of paths is limited
  - Large candidate set => high verification cost

gIndex
[Yan et al., SIGMOD'04]

- First work using pattern mining to do graph indexing
- Motivation: paths lose structural info =>

  filtering not effective enough =>

  use subgraphs to improve filtering
- Main idea: filtering by discriminative frequent subgraphs

gIndex [Yan et al., SIGMOD'04]

- Discriminative frequent subgraph
  - F: the set of frequent subgraphs in D
  - g is a discriminative frequent subgraph wrt F if

    $g \in F$ and $|D_g| << |\cap_{f \in F \wedge f \subset g} D_f|$
- Size-increasing support => reduce the size of F

gIndex
[Yan et al., SIGMOD'04]

- Index construction
  - Mine the set of discriminative frequent subgraphs, F, with a size-increasing support
- Query processing
  - Filtering:
    - Enumerate subgraphs of q, up to a size limit
    - Filter by inclusion logic: $C = \cap_{f \subseteq q \wedge f \in F} D_f$
  - Verification:
    - Test sub-Iso between q and each $g \in C$

gIndex [Yan et al., SIGMOD'04]

- Strengths
  - Subgraph features achieve better filtering than path features
  - Discriminative frequent subgraphs effectively eliminate redundancy in the feature set
- Limitations
  - Verification always needed: $|C| \geq |ans|$

TreePi [Zhang et al., ICDE'07]

- Motivation:
  - Many real graph datasets are tree-like
  - Trees are easier to manipulate than graphs
  - Trees retain more structural info than paths
- Main idea:
  - Filtering by discriminative frequent subtrees
  - Fast sub-Iso testing by measuring distance between tree centers
    - Tree center: by repeatedly removing leaves in a tree until a center node/edge remains

TreePi [Zhang et al., ICDE'07]

- Strengths
  - Lower indexing cost than subgraph approach
  - The use of tree center distance further reduces candidate set size and speeds up sub-Iso test
- Limitations
  - Filtering power of trees may be limited
  - Verification always needed: $|C| \geq |ans|$

Tree + Δ [Zhao et al., VLDB'07]

- Motivation:
  - Trees alone are not enough => need the help of some subgraphs on demand
- Main idea:
  - Filtering by frequent subtrees + on-demand discriminative subgraphs
  - Select on-demand a small set of graph-features F_g, where the filtering power of a graph-feature $f \in F_g$ is estimated from f's subtree-features

Tree + Δ [Zhao et al., VLDB'07]

- Strengths
  - Achieve similar filtering power of graph-features without costly graph mining => low indexing cost
- Limitations
  - Low indexing cost but query performance is bounded by that of using graph-features
  - On-demand graph-feature selection incurs extra query cost

Other Indexes using Features

- FG-index [Cheng et al., SIGMOD'07]: frequent subgraphs
- QuickSI [Shang et al., VLDB'08]: frequent subtrees

Representative Work

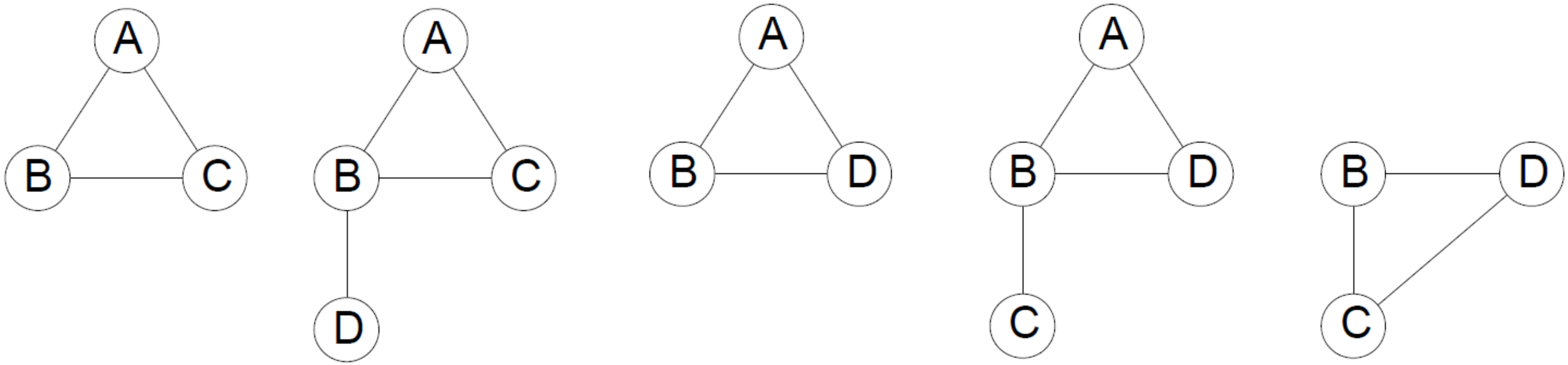
- Feature-based approach
- Closure-based approach
  - C-tree [He and Singh, ICDE'06]
  - Others: FG-index
- Verification-free approach
- Coding-based approach
- Fast sub-Iso approach

C-tree
[He and Singh, ICDE'06]

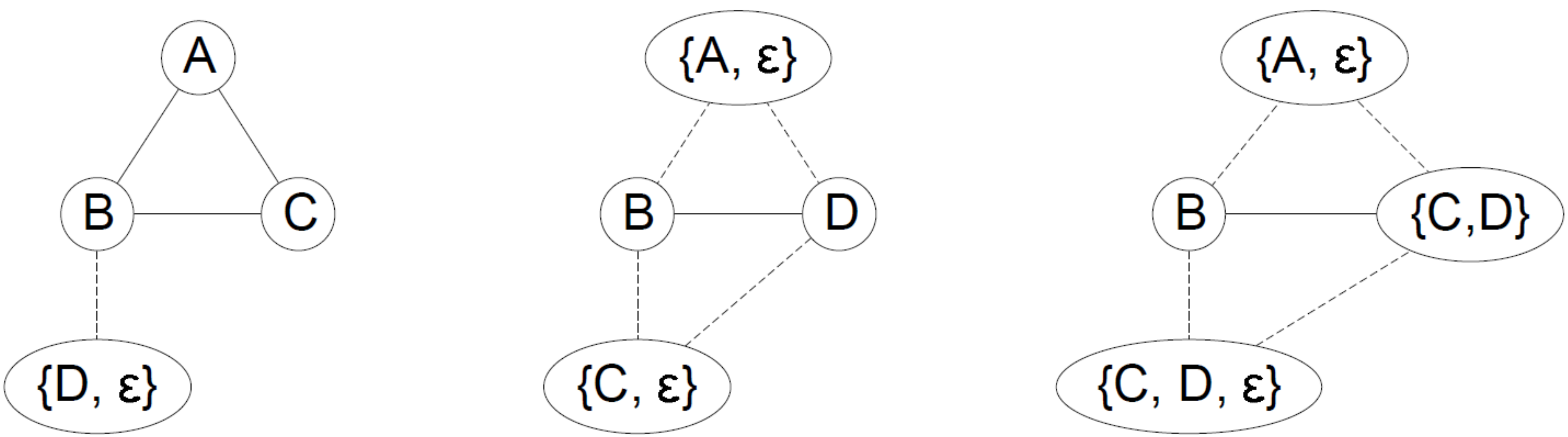
- First closure-based graph index
- Motivation:
  - Sub-structure features may still lose information of the original graphs
  - Use information of original graphs instead (to build an index tree)
- Main idea: an R-tree like graph index built on graph closures

C-tree
[He and Singh, ICDE'06]

- Closures
  - Vertex/edge closure: a set of vertices/edges => a single generalized vertex/edge
  - Graph closure: a set of graphs => a structural union of the graphs into a supergraph by some mapping, where common vertices/edges defined by vertex/edge closure



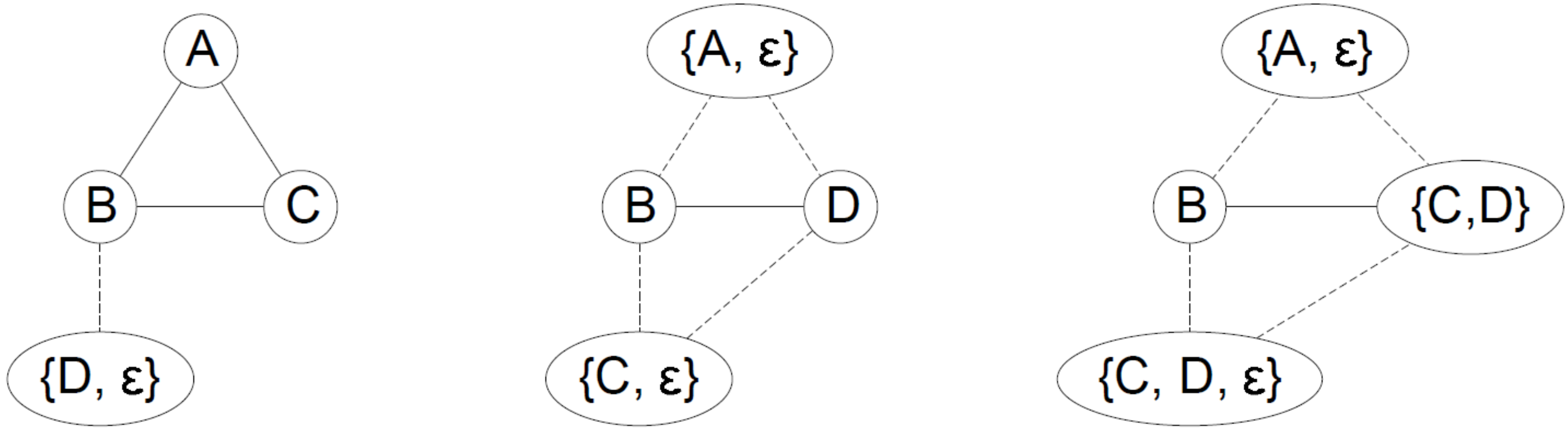
G_1 G_2 G_3 G_4 G_5



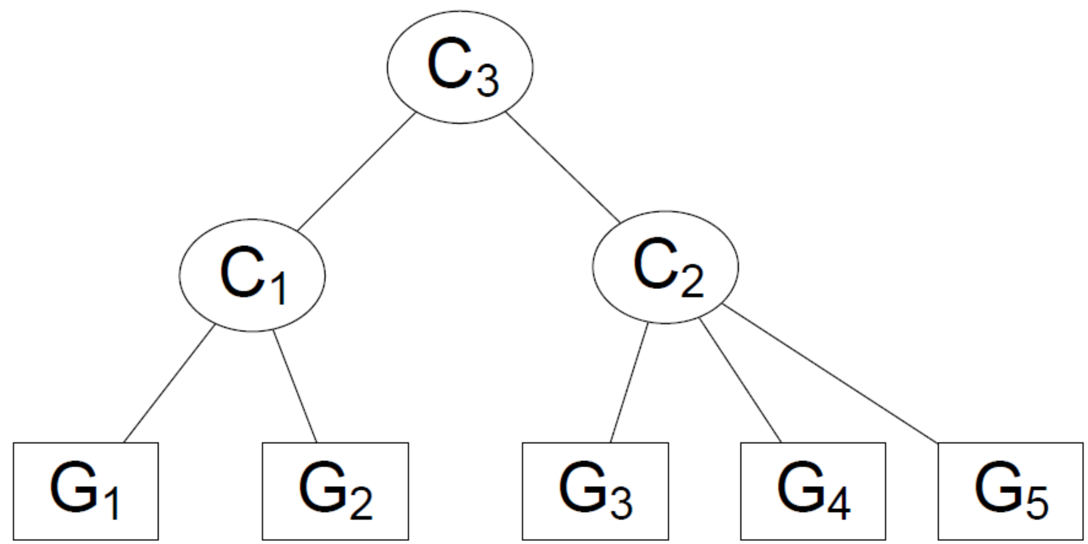
C_1=closure(G_1,G_2) C_2=closure(G_3,G_4,G_5) C_3=closure(C_1,C_2)

C-tree
[He and Singh, ICDE'06]

- Index construction
  - Construct an R-tree like index tree, C-tree, where each node is a closure of its children
  - Operations (e.g., insert, delete) of a C-tree similar to that of an R-tree



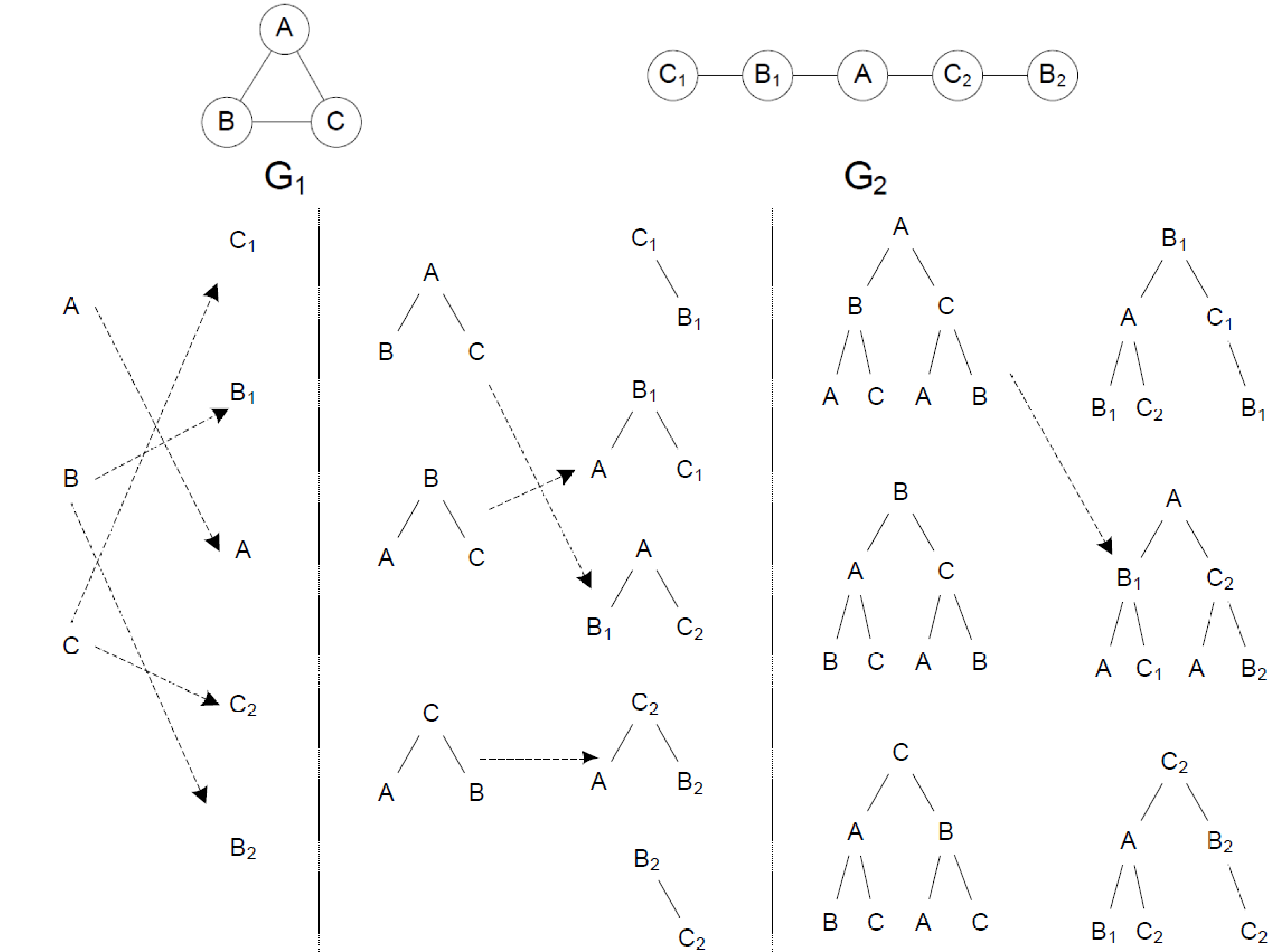
C_1=closure(G_1,G_2) C_2=closure(G_3,G_4,G_5) C_3=closure(C_1,C_2)



C-tree

C-tree [He and Singh, ICDE'06]

- Pseudo subgraph isomorphism
  - Given two graphs g_1 and g_2, for each node in each graph, grow a tree by BFS for n-steps
  - Approximate sub-Iso by matching the trees between the two graphs



G_1 is pseudo sub-isomorphic to G_2 at Step 3

C-tree
[He and Singh, ICDE'06]

- Query processing
  - Filtering:
    - Traverse the C-tree, filter out all nodes g if q is not pseudo sub-Iso to g
    - But if q is pseudo sub-Iso to g:
      - If g is not a data graph, visit all g's children
      - If g is a data graph, add g to C
  - Verification:
    - Test sub-Iso between q and each $g \in C$

C-tree [He and Singh, ICDE'06]

- Strengths
  - Support both subgraph and similarity queries
  - R-tree like structure
- Limitations
  - Verification always needed: $|C| \geq |ans|$

Other Indexes using Closure

- FG-index [Cheng et al., SIGMOD'07]:
  - A node in the FG-index tree represents a cluster of frequent subgraphs and can be regarded as a closure

Representative Work

- Feature-based approach
- Closure-based approach
- Verification-free approach
  - FG-index/FG*-index [Cheng et al., SIGMOD’07/TODS’09]
  - GDIndex [Williams et al., ICDE’07]
- Coding-based approach
- Fast sub-Iso approach

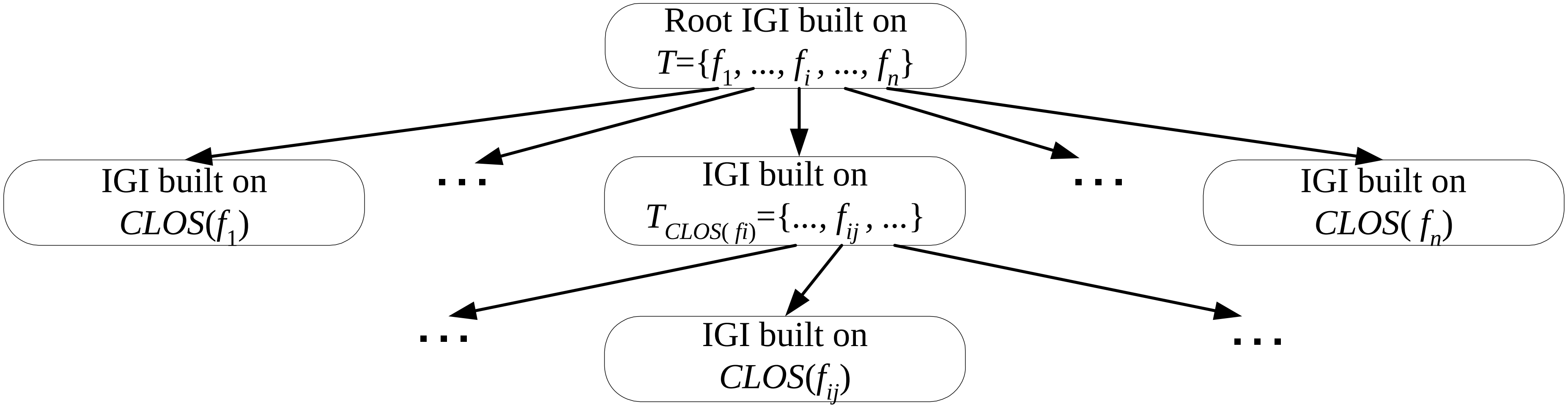
FG-index [Cheng et al., SIGMOD'07]

- First work proposes the concept of verification-free
- Motivation: filtering-and-verification approach requires at least $|C| \geq |ans|$ sub-Iso tests
- Main idea:
  - Answer an important subset of queries directly without verification
  - Answer the remaining queries with minimal verification

FG-index [Cheng et al., SIGMOD'07]

- Index construction
  - Mine the set of frequent subgraphs, F
  - Cluster F and organize it as an index tree, each node is a cluster
  - Recursively cluster a node (cluster) if it is too large => a multi-level index tree

FG-index



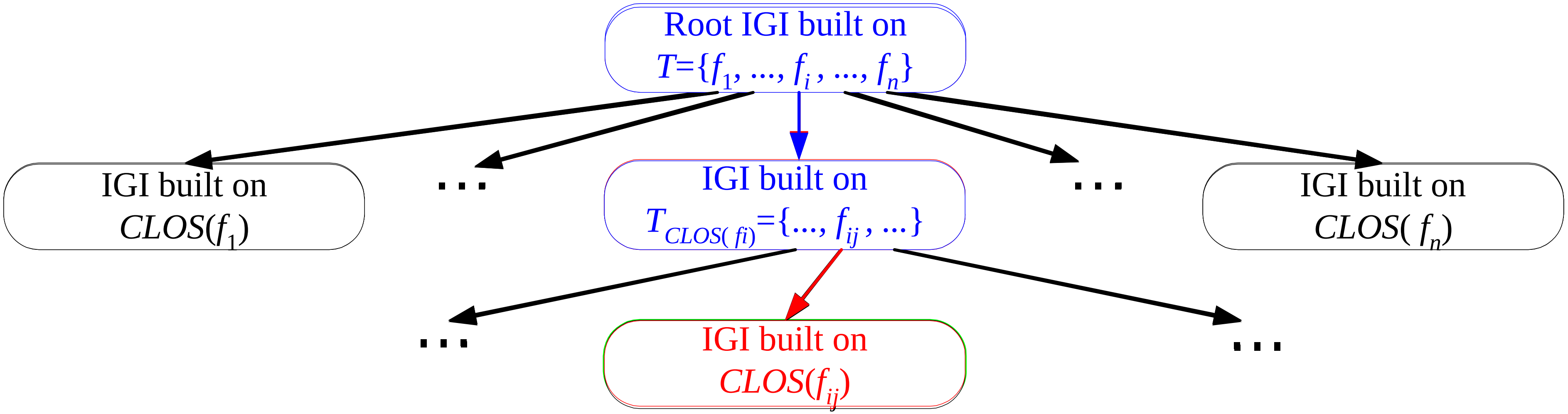
- FG-index is a multi-level index tree
- IGI: Inverted-Graph-Index built on a cluster of FGs

FG-index [Cheng et al., SIGMOD’07]

- Query processing
  - If q is a frequent subgraph (FG)
  - If q is not an FG

Query Processing

- When q is an FG



- Return query answer directly without any verification

FG-index
[Cheng et al., SIGMOD'07]

- When q is not an FG
  - Filtering-and-verification:
    - Find discriminative subgraphs, S, of q in FG-index
    - Filter by inclusion logic: $C = \cap_{f \in S} D_f$
    - Verification: test sub-Iso between q and each $g \in C$

FG-index [Cheng et al., SIGMOD'07]

- Strengths
  - Verification-free for answering FG-queries (i.e., queries that have the largest verification cost)
- Limitations
  - FG-index may have a high index-probing cost if F is too big
  - Non-FG queries are still answered by the filtering-and-verification framework

FG*-index [Cheng et al., TODS'09]

- A feature-index: to facilitate efficient index-probing in FG-index
- An FAQ-index: to answer non-FG queries without verification in general

GDIndex [Williams et al., ICDE'07]

- Motivation: graphs in many applications are small
- Main idea:
  - Hash all subgraphs of all graphs in the database
  - Match a query by hashing
  - Focus on graphs with limited sizes

GDIndex [Williams et al., ICDE'07]

- Strengths
  - No verification for any query
- Limitations
  - Not suitable for applications with large graphs

Representative Work

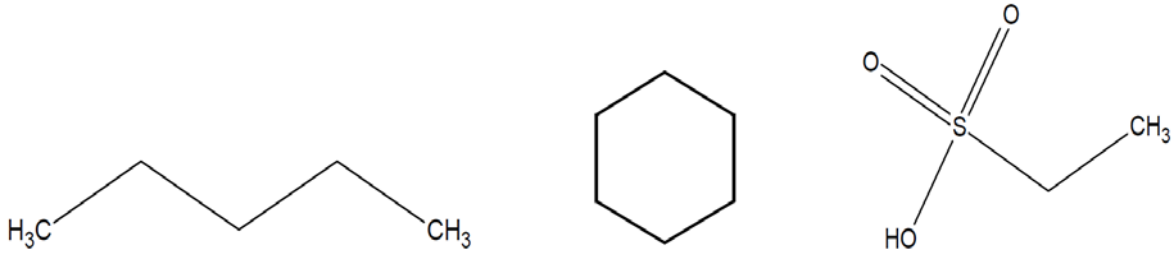
- Feature-based approach
- Closure-based approach
- Verification-free approach
- Coding-based approach
  - GString [Jiang et al., ICDE’07]
  - GCoding [Zou et al., EDBT’08]
- Fast sub-Iso approach

GString [Jiang et al., ICDE'07]

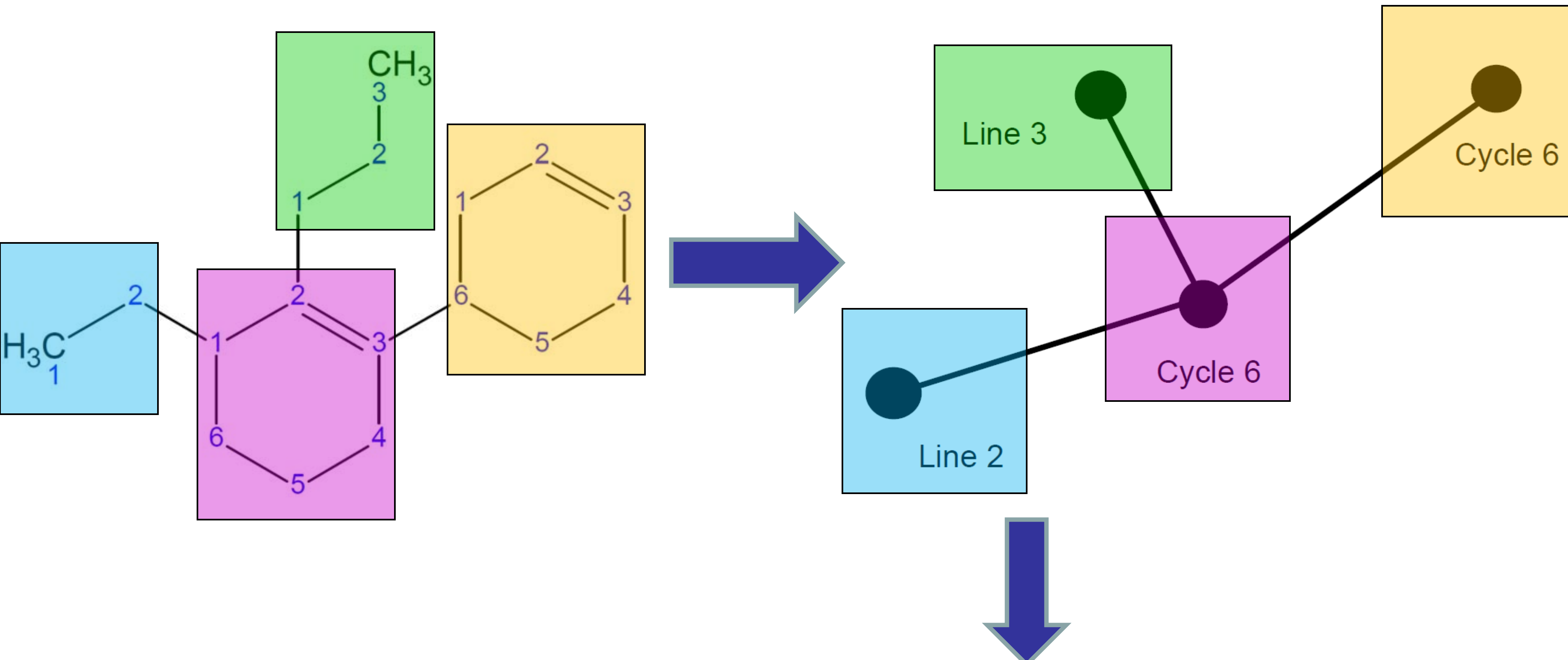
- Motivation: existing feature-based approaches do not consider semantics of structures
- Main idea:
  - Encode graphs into strings, using semantics of sub-structures
  - Transform subgraph query processing into string matching

GString [Jiang et al., ICDE'07]

- Index construction
  - Semantics of basic graph structures: line, cycle, star
  - Use a grammar to convert a graph into a string consisting of its basic structures
  - Construct a suffix tree for all graph strings



(a) Line (b) Cycle (c) Star



GString: Line 2 … Cycle 6 … Line 3 … Cycle 6 …

GString
[Jiang et al., ICDE'07]

- Query processing
  - Encode q as a string
  - Filter out false results by matching q with the suffix tree
  - Verify each matching string (of a graph g) by testing sub-Iso between q and g

GString
[Jiang et al., ICDE'07]

- Strengths
  - Index considers semantics of sub-structures
- Limitations
  - Verification always needed: $|C| \geq |ans|$

GCoding [Zou et al., EDBT'08]

- Motivation: spectral graph theory pruning rules have shown to be effective for processing twig queries in XML
- Main idea:
  - Use spectral graph coding to encode the structure of a graph into a numerical space
  - Encode q and match q by comparing graph codes

GCoding [Zou et al., EDBT'08]

- Strengths
  - Graph codes easy to update => support frequent updates
- Limitations
  - Verification always needed: $|C| \geq |ans|$

Representative Work

- Feature-based approach
- Closure-based approach
- Verification-free approach
- Coding-based approach
- Fast sub-Iso approach
  - QuickSI [Shang et al., VLDB'08]
  - Others: C-tree, TreePi

QuickSI [Shang et al., VLDB'08]

- Motivation:
  - All existing works, except FG-index and GDIndex, adopt the filtering-and-verification framework
  - Verification cost dominates due to sub-Iso

QuickSI [Shang et al., VLDB’08]

- Main idea:
  - Improve the sub-Iso test in the verification step
  - Reduce branch-and-bound in Ullman’s sub-Iso algorithm, by an effective search order based on
    - The frequencies of vertices/edges in the underneath graph database
    - The topological info of the graphs

QuickSI [Shang et al., VLDB'08]

- Strengths
  - Reduce verification cost by a fast sub-Iso algorithm
- Limitations
  - Verification always needed: $|C| \geq |ans|$

Other Fast Sub-Iso Approach

- TreePi [Zhang et al., ICDE'07]: use tree center distance constraint
- C-tree [He and Singh, ICDE'06]: pseudo subgraph isomorphism

	Feature-based	Closure-based	Verification-free	Coding-based	Fast sub-Iso
GraphGrep [Shasha et al., PODS'02]	X				
gIndex [Yan et al., SIGMOD'04]	X				
C-tree [He and Singh, ICDE'06]		X			X
FG-index [Cheng et al., SIGMOD'07]	X	X	X		
GString [Jiang et al., ICDE'07]				X	
TreePi [Zhang et al., ICDE'07]	X				X
GDIndex [Williams et al., ICDE'07]			X		
Tree+Δ [Zhao et al., VLDB'07]	X				
GCoding [Zou et al., EDBT'08]				X	
QuickSI [Shang et al., VLDB'08]	X				X

Conclusions on Subgraph Query Processing

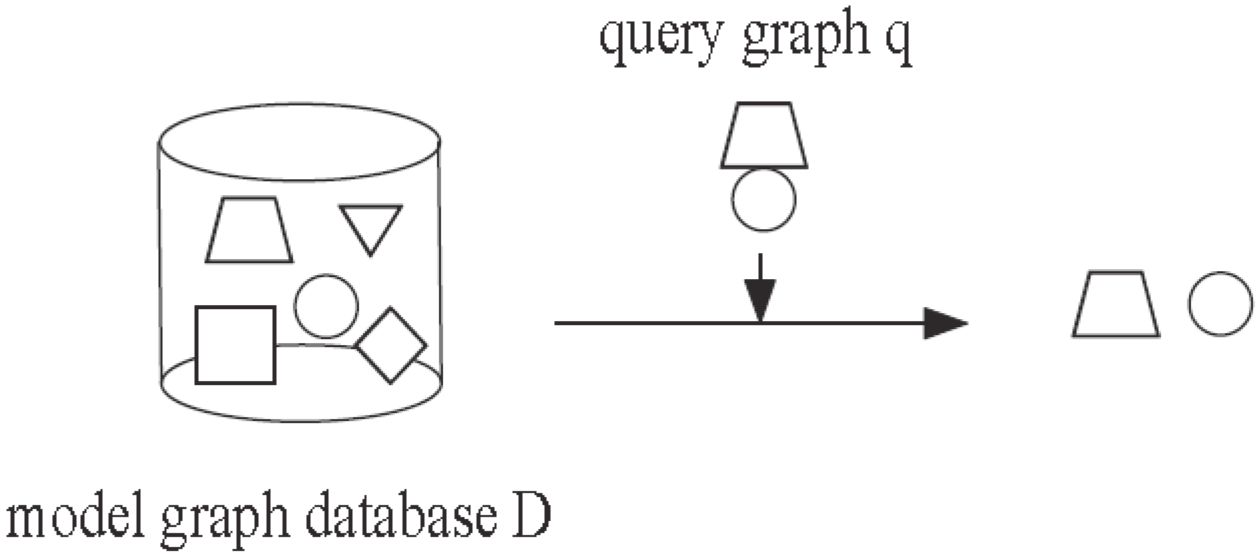
- Five different approaches (roughly)
  - Feature-based approach: GraphGrep, gIndex, TreePi, Tree+Δ, FG-index, QuickSI
  - Closure-based approach: C-tree, FG-index
  - Verification-free approach: FG-index, GDIndex
  - Coding-based approach: GString, GCoding
  - Fast sub-Iso approach: QuickSI, C-tree, TreePi
- Overall performance
  - Strengths and limitations of each work briefly discussed
  - Performance depends on applications and individual focuses, no clear winner

Tutorial Coverage

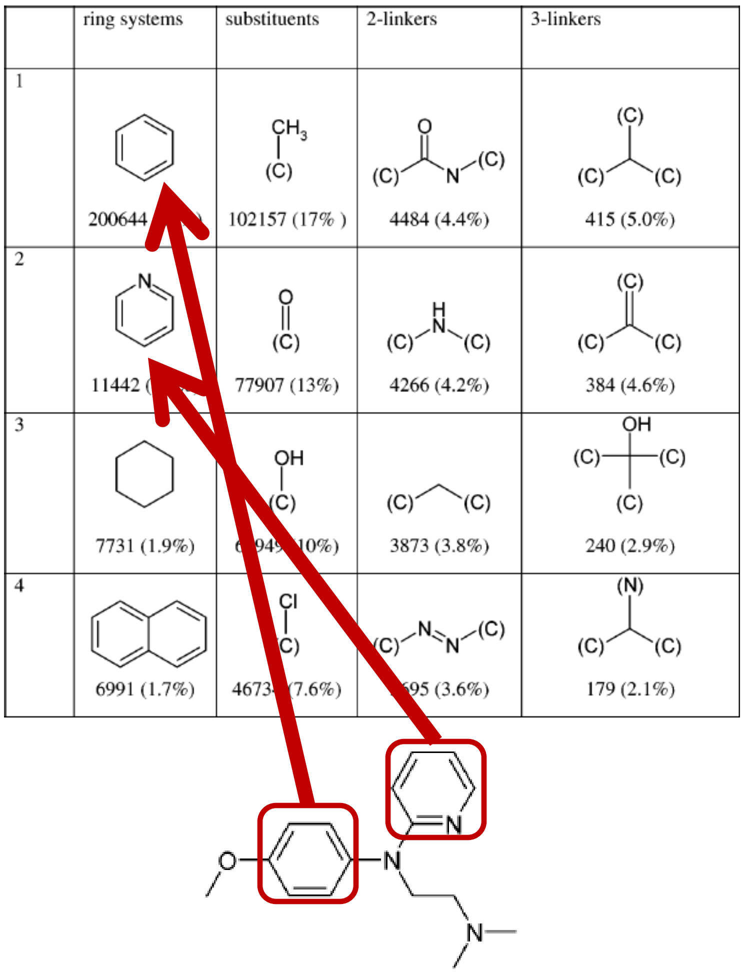
- Subgraph queries
- Supergraph queries
- Similarity queries

Supergraph Query Processing

- Counterpart of subgraph query processing
- Problem
  - Given a graph database D and a graph query q
  - Find all graphs g in D s.t. q is a supergraph of g



Many Applications



Chemical Descriptor Identification
[Lameijer et al. 2006]

Object Recognition
(from SIFT project, Standford)

Challenges

- **Problem complexity: NP-complete**
  - Same as subgraph query
- **Existing feature-based indexes for subgraph queries are not applicable**
  - Inclusion logic for subgraph query
    - If $f \subseteq q$ and $f \nsubseteq g$, then $q \nsubseteq g$
  - Exclusion logic for supergraph query
    - If $f \nsubseteq q$ and $f \subseteq g$, then $q \nsupseteq g$
  - Need to design different feature selection mechanisms

Supergraph Query Processing

- **Representative work**
  - cIndex [Chen et al., VLDB'07]
    - Feature-based approach
  - GPTree [Zhang et al., EDBT'09]
    - Feature-based approach
    - Fast sub-Iso approach

cIndex [Chen et al., VLDB'07]

- **First work on supergraph query processing**
- **Basic framework**

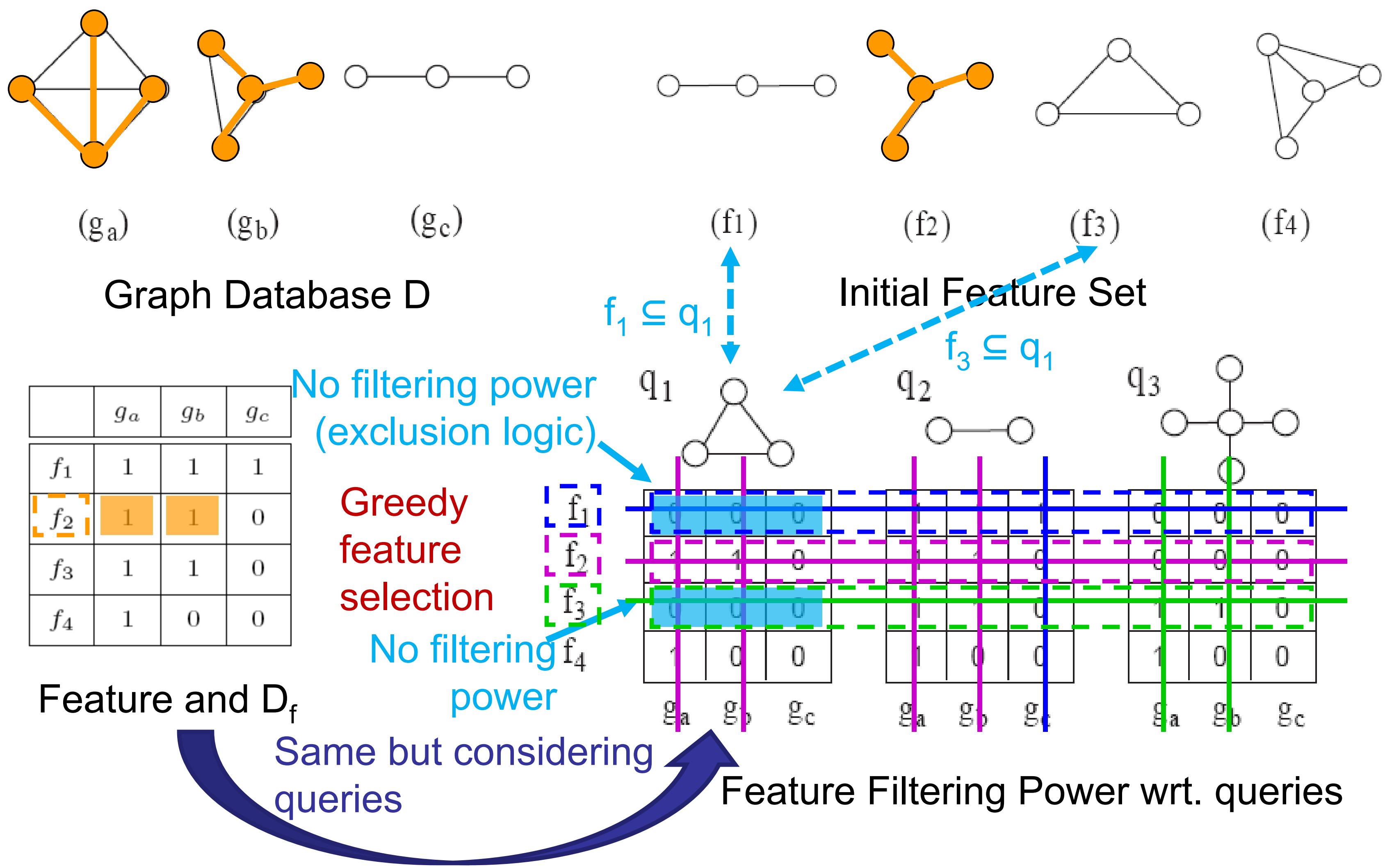
1. Off-line index construction
   - Generate and select a feature set F
   - For $f \in F$, store $D_f = \{ g \mid f \subseteq g \wedge g \in D\}$
2. Filtering
   - Check if $f \subseteq q$ for each $f \in F$ (by sub-Iso test)
   - Compute a candidate set C by exclusion logic

     $$C = D - \bigcup_{f \nsubseteq q \wedge f \in F} D_f$$
3. Verification
   - Verify if $q \supseteq g$, for each $g \in C$ (by sub-Iso test)

Feature Selection

- Generate an initial feature set F_0 by FG mining
- Select a subset F of F_0 with the best filtering power (D_f is large and $f \nsubseteq q$)
- Use a query log to measure the feature filtering power

Greedy Feature Selection



	g_a	g_b	g_c
f_1	1	1	1
f_2	1	1	0
f_3	1	1	0
f_4	1	0	0

GPTree [Zhang et al., EDBT'09]

- **Main idea**
  - Improve query performance in two aspects
    - Select *significant* features → feature-based approach
    - Organize data graphs/features to reduce sub-Iso tests with q → Fast sub-Iso approach

Feature Selection

- **Large subgraphs are preferred as features**
  - Less likely to be contained by q $\rightarrow$ apply exclusion logic
  - If f $\subseteq$ f' and $D_f = D_{f'}$, select f' as a feature $\rightarrow$ prefer closed FGs
- **Significance metric δ of a subgraph f**

$$\delta(f) = \frac{|D_f|}{\boxed{|\bigcup_{f_i \in \mathcal{F}, f_i \supseteq f} D_{f_i}|}}$$

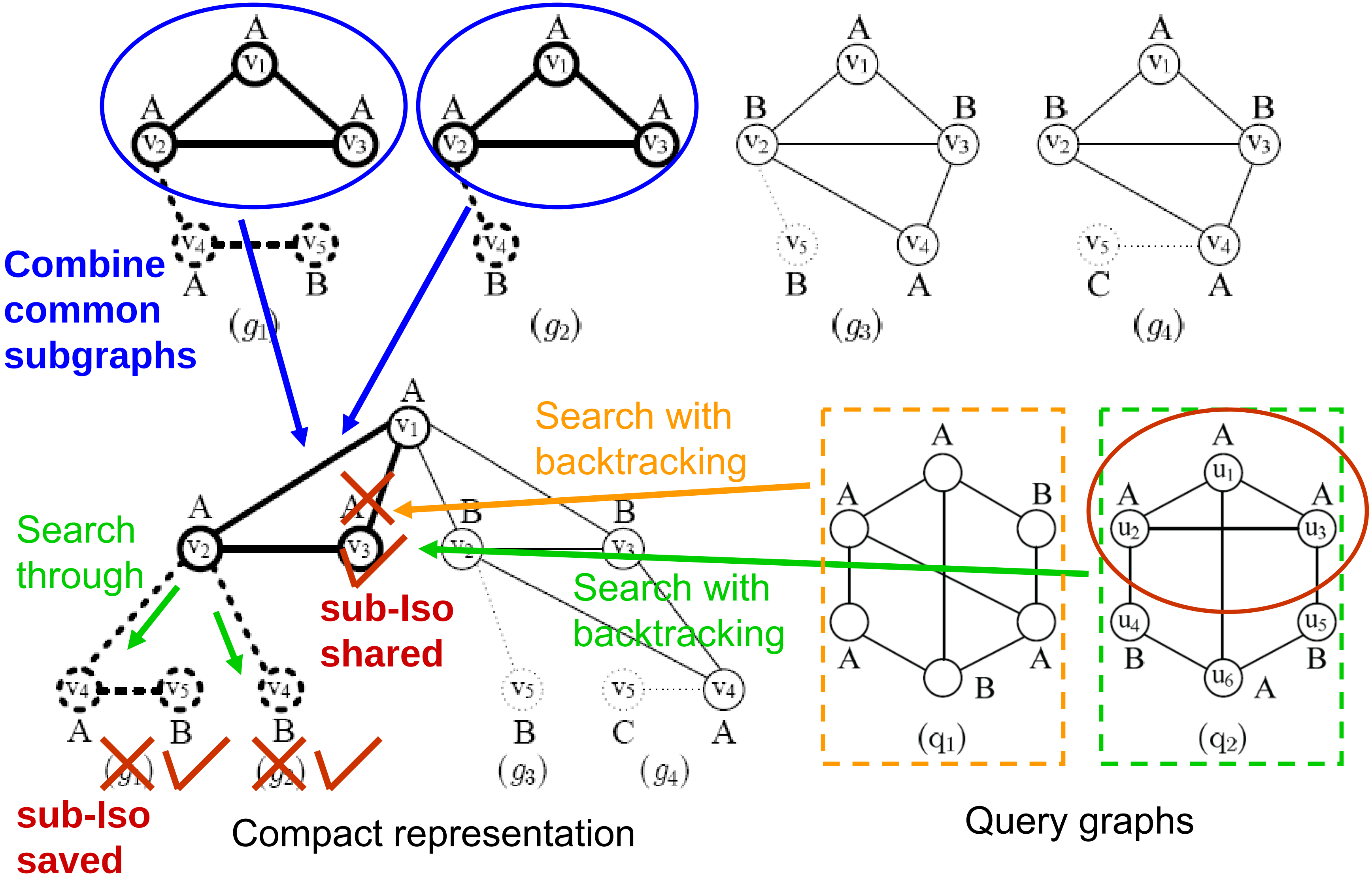
data graphs covered by f

How much more filtering power f can bring in

data graphs already covered by current features

- **Feature selection**
  - Mine CFGs from D; remove those with $\delta(f) < \delta_{min}$
  - Proceed from large subgraphs to small ones

Organize Data Graphs / Features



Tutorial Coverage

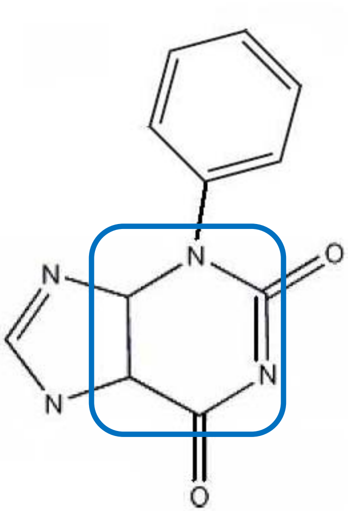
- Subgraph queries
- Supergraph queries
- Similarity queries

Similarity Search

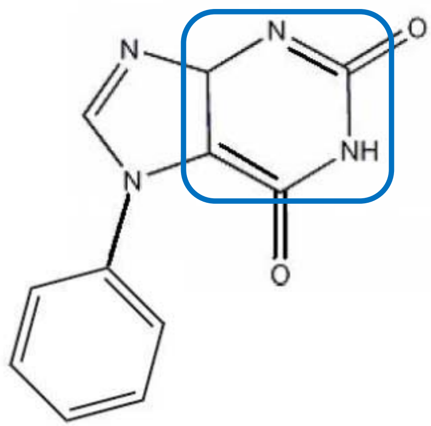
- **Why similarity search?**
  - Data may not be error-free
  - Application need
    - object recognition, protein-ligand docking, etc.
- **Two categories**
  - Structural similarity search
    - Find graphs with structure similar to q
  - Distribution similarity search
    - Find graphs with occurrence distribution similar to q

Structural Similarity Search

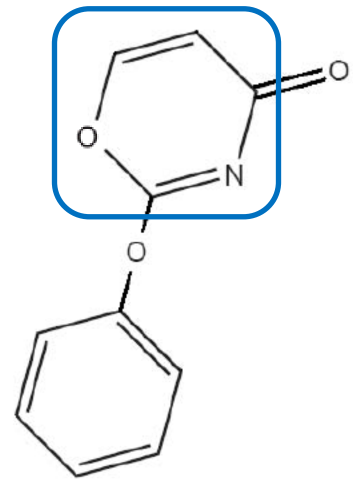
- **Find graphs that have similar structure to q wrt. a similarity measure**
  - sim(g, q) $\geq \delta$



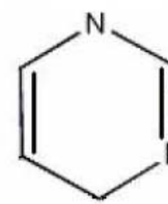
Graph A



Graph B



Graph C



Query graph q

Graph Database

Structural Similarity Search

- **Three types based on query characteristic**
  - q is a full structure of data graphs
  - q is a subgraph of data graphs
  - q is a supergraph of data graphs

Query Type	Full Structure	Subgraph Query	Supergraph Query
Exact Match		gIndex C-tree FG-index QuickSI …	cIndex GPTree
Structural Similarity	**RASCAL**	**Grafil**	**SG-Enum**

RASCAL [Raymond et al., CJ'02]

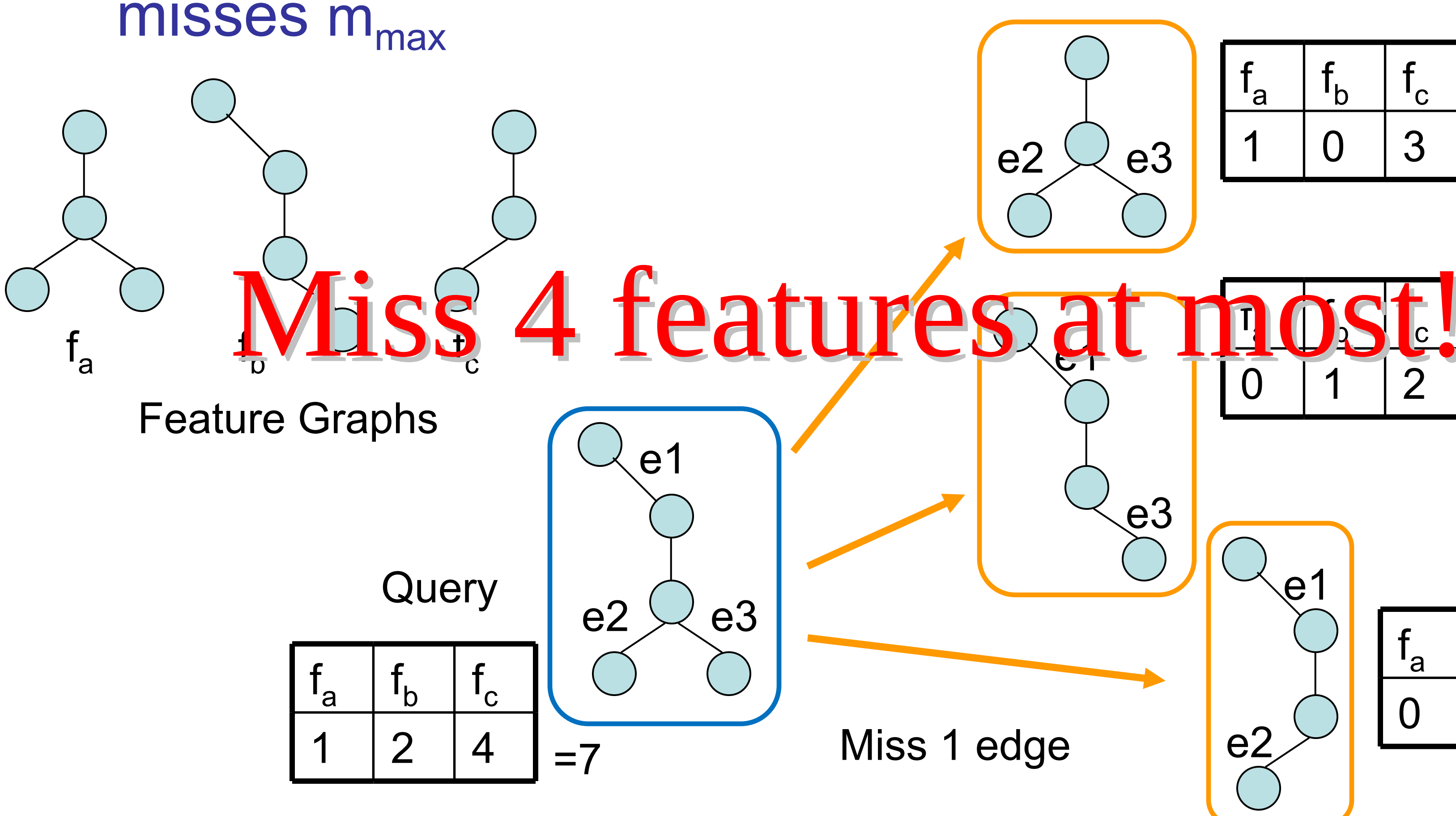
- **Full structure similarity search**
- **Similarity measure**
  - Relative size of the maximum common edge subgraph (MCES)
- **Main idea**
  - Filtering
    - Remove very dissimilar data graphs
    - Two-tiered upper bound pruning
  - Verification
    - Test whether sim(g, q) $\geq \delta$
    - Compute MCES of for each remaining g and q

RASCAL – Filtering

- **First tier**
  - Consider vertex label and vertex degree
  - Match vertex arbitrarily by the same label and degree
  - A loose upper bound of sim(g, q)
- **Second tier**
  - Further consider edge label
  - Instead of matching by vertex degree, match by compatible edges
  - A tighter upper bound but more costly

Grafil [Yan et al., SIGMOD'05]

- **Subgraph similarity search: q is smaller**
- **Main idea:** transform edge misses k to feature misses m_{max}



Miss 4 features at most!

Query:

f_a	f_b	f_c
1	2	4

=7

Query missing e1:

f_a	f_b	f_c
1	0	3

=4

Query missing e2:

f_a	f_b	f_c
0	1	2

=3

Query missing e3:

f_a	f_b	f_c
0	1	2

=3

Feature-based Filtering

- **How to use the feature misses m_{max}**
  - $m_{max} = 4$

feature misses =(1-0)+(2-0)+(4-2) = 5 > m_{max}

feature misses =(1-1)+(2-0)+(4-3) = 3 < m_{max}

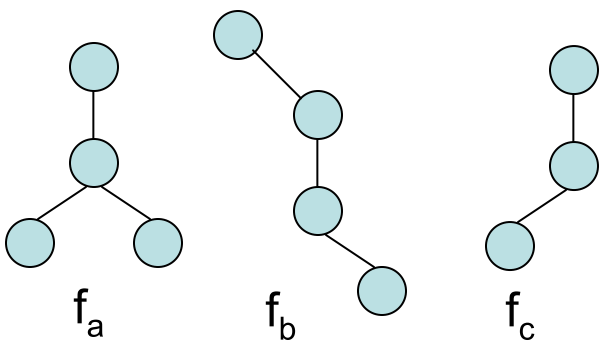
feature misses =(1-0)+(2-1)+(4-4) = 2 < m_{max}

feature misses =(1-1)+(2-0)+(4-0) = 6 > m_{max}

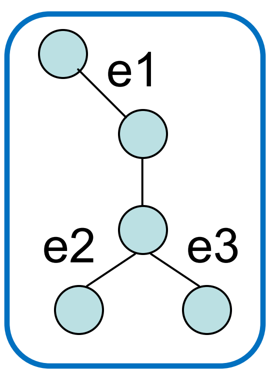
	f_a	f_b	f_c	
g1	0	0	2	✗
g2	1	0	3	✓
g3	0	1	4	✓
g4	1	0	0	✗
q	1	2	4	

How to Calculate Feature Misses?

- **Enumerating all relaxed queries is expensive**
- **Classic set k-cover problem**
  - k: the number of missing edges in q
  - m_{max}: max number of features covered by k edges



q



Until k edges are selected

	f_a	f_{b1}	f_{b2}	f_{c1}	f_{c2}	f_{c3}	f_{c4}
e1	0	1	1	1	0	0	0
e2	1	1	0	0	1	0	1
e3	1	0	1	0	0	1	1

Edge-Feature Matrix

SG-Enum [Shang et al., ICDE'10]

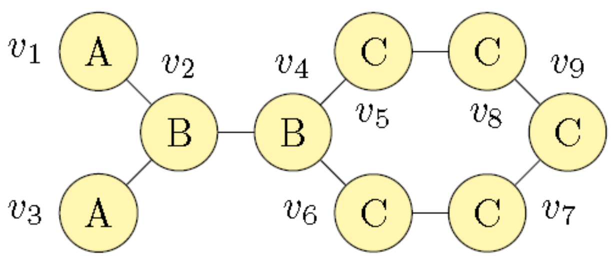
- **Supergraph similarity search: q is larger**
- **Similarity measure**
  - Maximum common subgraph (MCS)
  - Given query q and data graph g

$$dis(q, g) = |g| - |mcs(q, g)|$$

- **Problem Definition**
  - Find all data graphs g in D with $dis(q, g) \leq \sigma$

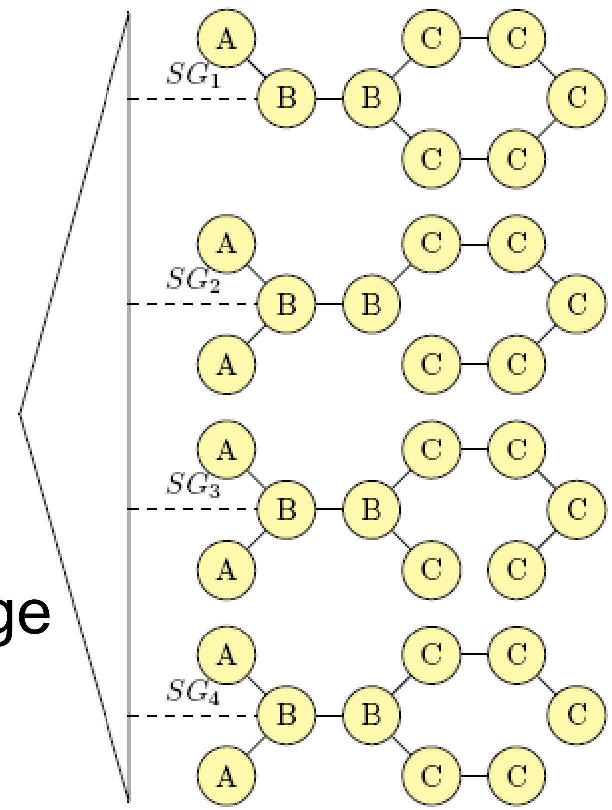
σ-Missing Subgraphs

- **Main idea:** relax data graph g instead of q
  - Allow g to miss σ edges



g

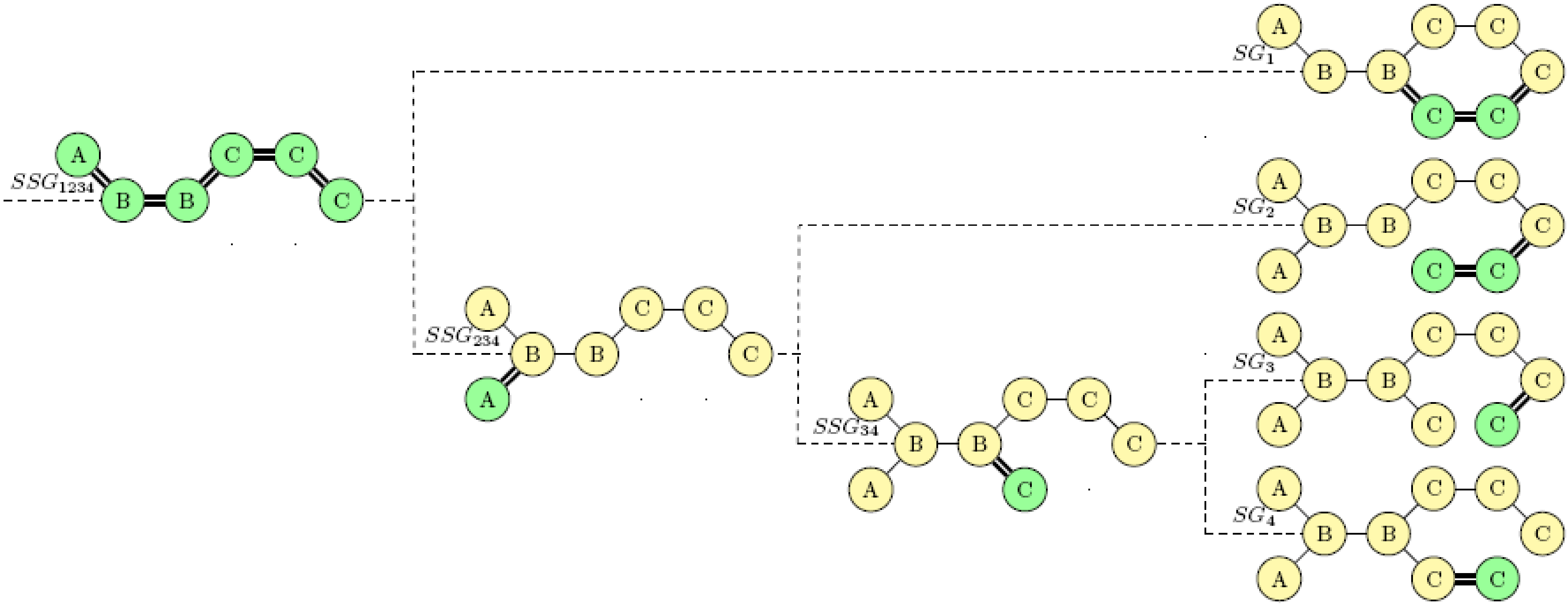
miss 1 edge



g's σ-missing subgraphs

Query Processing

- **SG-Enum index**
  - Organize σ-missing subgraphs in a tree
- **Search q on the index by testing sub-Iso**
  - g is an answer graph iff at least one leaf node s ⊆ q



Tutorial Coverage

- Subgraph queries
- Supergraph queries
- Similarity queries
  - Structural similarity queries
  - Distribution similarity queries

Distribution Similarity Search

- **Occurrence of a subgraph in a data graph: a boolean variable**
- **Distribution similarity search**
  - Find subgraphs that have similar occurrence distribution to q wrt. a correlation measure

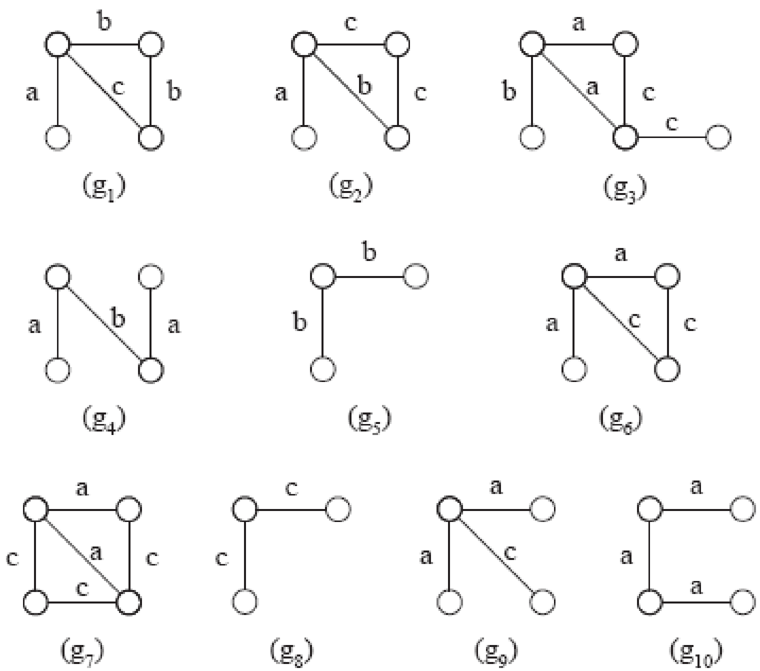


Fig. (a): Graph database

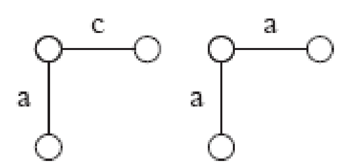


Fig. (b): Two subgraphs

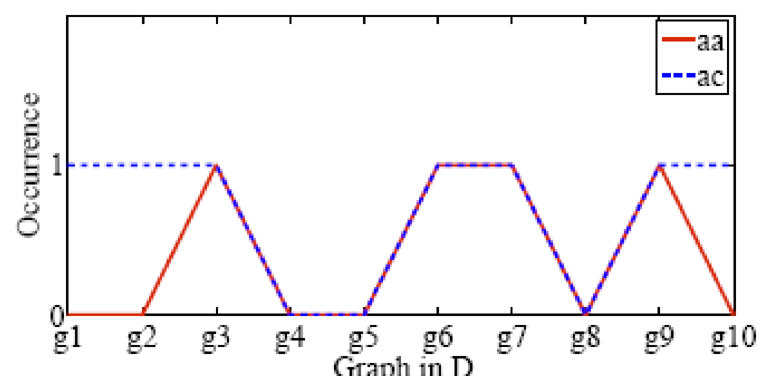
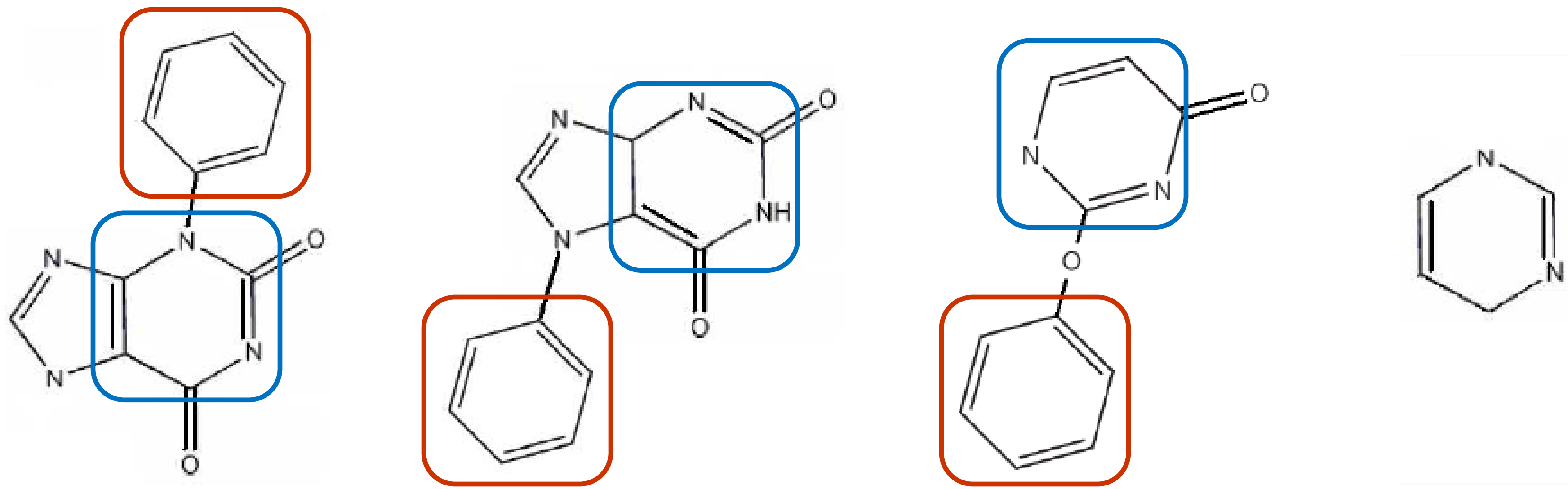


Fig. (c): Subgraph occurrence distribution

Why Distribution Similarity?

- **Subgraphs with similar distributions**
  - Capture the underlying occurrence dependency
  - May imply the same hidden property
  - May be structurally similar / dissimilar



(a) Graph A (b) Graph B (c) Graph C (d) Query

Distribution Similarity Search

- **Challenges**
  - Huge search space: not linear in # of data graphs, but linear in # of subgraphs of data graphs
- **Representative work**
  - CGSearch [Ke et al., KDD'07]
    - Threshold-based approach
  - TopCor [Ke et al., SDM'09]
    - Top-k
  - FCP-Miner [Ke et al., ICDM'09]
    - Discover all distribution-similar subgraph pairs

CGSearch [Ke et al., KDD'07]

- **Correlation measure: Pearson's coefficient**
  - Measure the departure of two variables from independence
  - *supp(g)* represents the occurrence probability of a graph g

$$\phi(g_1, g_2) = \frac{supp(g_1, g_2) - supp(g_1)supp(g_2)}{\sqrt{supp(g_1)supp(g_2)(1 - supp(g_1))(1 - supp(g_2))}}$$

- **Problem**
  - Given a database D, a query q, and a threshold θ
  - Find all subgraphs g in D with $\phi(q, g) \geqslant \theta$

CGSearch

- **Basic framework**
  - Candidate generation and filtering
    - Transform the search space from D to D_q
    - Use heuristic rules to further prune false-positive candidates
  - Verification
    - Compute $\phi(q, g)$ for each g in the candidate set
    - Return those g with $\phi(q, g) \geqslant \theta$ as answers

Candidate Generation

- **Candidate generation**
  - Derive a lower bound for the joint support in D_q

$$supp(q, g; \mathcal{D}_q) \geq \frac{1}{\theta^{-2}(1 - supp(q)) + supp(q)}$$

  - Generate candidates from D_q by FG-mining with the above bound
- **Advantages**
  - Significant reduction in search space: $D_q << D$
  - Efficient candidate generation

Candidate Filtering

- **Heuristic 1**
  - All supergraphs of q in the candidate set are answers for sure
  - Include answers directly without verification
- **Heuristic 2**
  - If $\phi(q, g) < \theta$, all subgraphs of g with the same support can be safely pruned
  - Remove false-positives and save unrewarding verification

TopCor [Ke et al., SDM'09]

- **Problem**
  - Given a database D, a query q, and an integer k
  - Find top-k subgraphs g in D with the highest ϕ (q, g)
- **Why top-k?**
  - Circumvent the need for a user-specified correlation threshold θ
  - Allow a user to directly control the number of patterns discovered
- **Challenges**
  - Inefficient to use CGSearch
  - Hard to find a connection between k and θ

TopCor

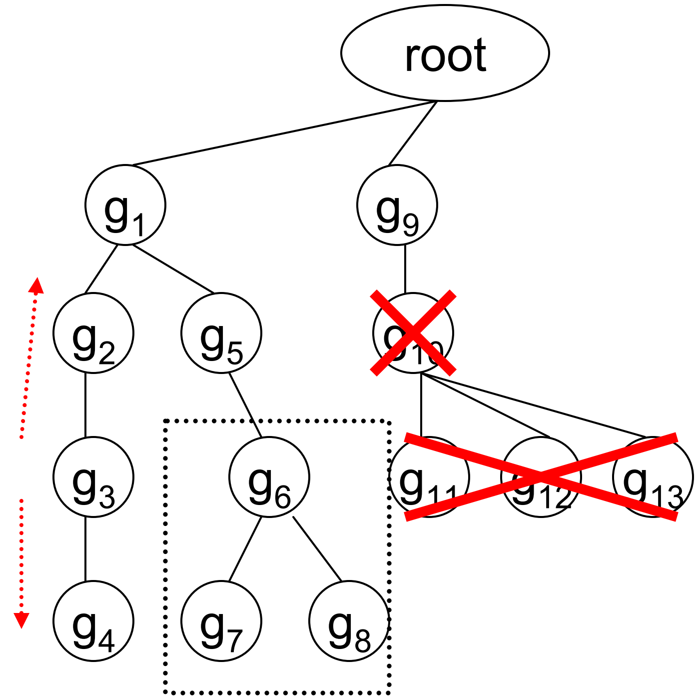
- **Main idea**
  - Mine subgraphs in D_q by growing a search tree T in a depth-first manner
  - Maintain a priority queue for current top-k results
  - When exploring T, apply three key techniques to direct the search to those highly correlated subgraphs

Key Techniques

- **T1: early correlation checking**
  - Identify an upper bound of $\phi(q, g)$ for a subgraph g
  - ϕ_{min}: minimum ϕ in the current priority queue
  - If upper($\phi(q, g)$) < ϕ_{min}, prune g
- **T2: Branch pruning**
  - upper($\phi(q, g)$) is anti-monotonic
  - If upper($\phi(q, g)$) < ϕ_{min}, prune all supergraphs of g
- **T3: Heuristic rules**
  - Rule 1: skip verification for supergraphs of q
  - Rule 2: first verify *closed* subgraphs
  - Rules 3-5: prune subgraphs/supergraphs of a verified g

TopCor Search Process

- Depth-first exploration …
- g_3 is a closed subgraph
- Verification on g_3 by Rule 2
- Pruning upward from g_3 by Rule 3 and downward by Rules 4-5
- g_5 is the query q
- Skip verification in g_5's branch by Rule 1
- upper(g_{10}) < ϕ_{min}, prune g_{10} by T1
- Prune branch of g_{10} by T2



Search Tree T

FCP-Miner [Ke et al., ICDM'09]

- **Problem**
  - Given a database D, a support threshold σ, and a correlation threshold θ
  - All pairs of subgraphs (f_1, f_2) such that

$$supp(f_1) \geqslant \sigma,\ supp(f_2) \geqslant \sigma, \text{ and } \phi(f_1, f_2) \geqslant \theta$$

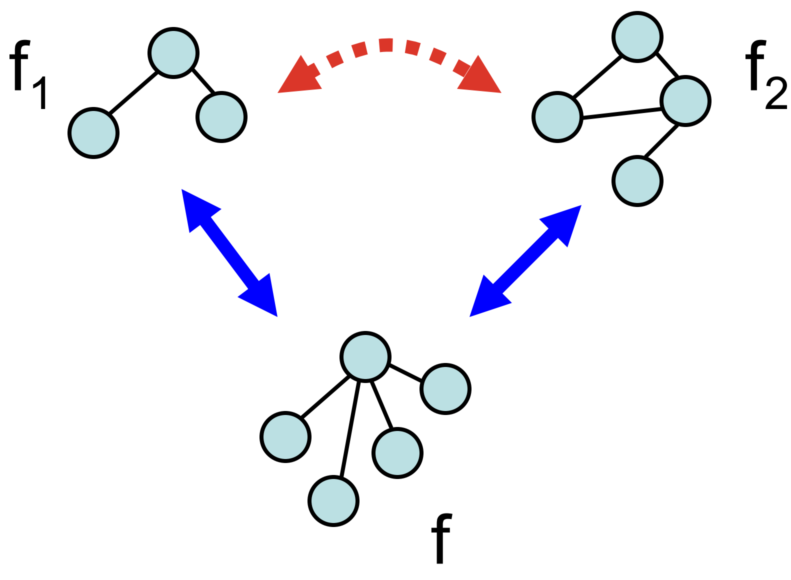
- **Why all pairs?**
  - A query graph may not be available
  - Applications need to investigate all possibilities (drug design)
- **Challenges**
  - Feeding every subgraph in D to CGSearch is infeasible

FCP-Miner

- **Answer set of a frequent subgraph f**
  - $A_f = \{f' : supp(f') \geqslant \sigma, \phi(f, f') \geqslant \theta\}$
  - The set of subgraphs that form answer pairs with f
- **Main idea**
  - Compute exact answer sets for only a small number of FGs
  - Use these answer sets to approximate the answer sets of the remaining FGs

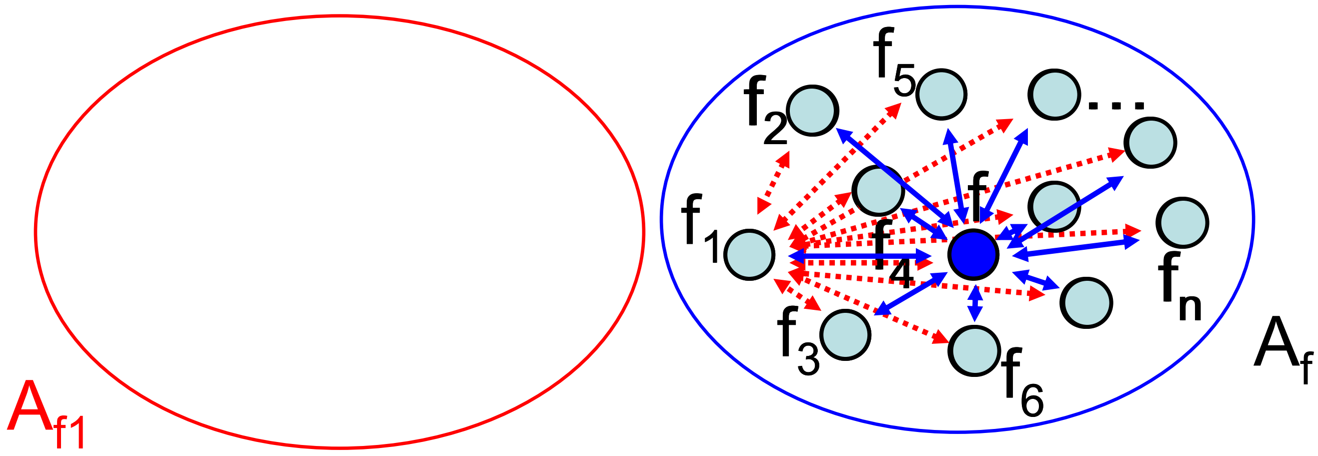
Correlation Property

- **Correlation tends to be "*transitive*"**
  - If f_1 and f_2 are both correlated to the same subgraph f, they are likely to be correlated as well



How to Use the Property?

- All subgraphs correlated to f are in A_f
- Consider a subgraph f_1 in A_f
- By the transitive property, f_1 is likely to be correlated with any other subgraph in A_f
- Approximate A_{f1} based on A_f
  - Skip obtaining the exact A_{fx}, $\forall f_x \in A_f$



Query Type	Full Structure	Subgraph Query	Supergraph Query
Exact		GraphGrep gIndex C-tree FG-index GString GDIndex Tree + △ GCoding QuickSI	cIndex GPTree
Structural Similarity	RASCAL	Grafil	SG-Enum
Distribution Similarity		CGSearch TopCor FCP-Miner	

Future Directions

- Imbalanced development of subgraph queries vs. supergraph/similarity queries
  - The later two are relatively new
  - Many technical aspects remain unexplored
- Scalability problem
  - Existing work evaluated on databases of < 1M graphs
  - Rapid growth in graph data (billions of graphs)
  - A hybrid approach that combines the strengths of existing work might be feasible
  - Disk-based index is another possible direction

Future Directions

- More sophisticated queries or knowledge discovery built upon these primitive queries
  - Aggregate query
  - Classification
- Subgraph/supergraph/similarity queries on other types of graph data
  - Sequential graph data
  - Evolving graph data
  - Uncertain graph data
  - Probabilistic graph data

References

- [Shasha et al., PODS'02] Shasha, D., Wang, J.T.L., Giugno, R.: Algorithmics and applications of tree and graph searching. In: PODS. (2002) 39–52
- [Yan et al., SIGMOD'04] Yan, X., Yu, P.S., Han, J.: Graph indexing based on discriminative frequent structure analysis. In: SIGMOD. (2004) 335–346
- [He and Singh, ICDE'06] He, H., Singh, A.K.: Closure-tree: An index structure for graph queries. In: ICDE. (2006) 38
- [Cheng et al., SIGMOD'07] Cheng, J., Ke, Y., Ng, W., Lu, A.: Fg-index: towards verification-free query processing on graph databases. In: SIGMOD. (2007) 857–872
- [Cheng et al., TODS'09] Cheng, J., Ke, Y., Ng, W.: Effective query processing on graph databases. ACM Trans. Database Syst. 34(1) (2009)
- [Jiang et al., ICDE'07] Jiang, H., Wang, H., Yu, P.S., Zhou, S.: Gstring: A novel approach for efficient search in graph databases. In: ICDE. (2007) 566–575
- [Zhang et al., ICDE'07] Zhang, S., Hu, M., Yang, J.: Treepi: A novel graph indexing method. In: ICDE. (2007) 966–975
- [Williams et al., ICDE'07] Williams, D.W., Huan, J., Wang, W.: Graph database indexing using structured graph decomposition. In: ICDE. (2007) 976–985

References

- [Zhao et al., VLDB'07] Zhao, P., Yu, J.X., Yu, P.S.: Graph indexing: Tree + delta >= graph. In: VLDB. (2007) 938–949
- [Zou et al., EDBT'08] Zou, L., Chen, L., Yu, J.X., Lu, Y.: A novel spectral coding in a large graph database. In: EDBT. (2008) 181–192
- [Shang et al., VLDB'08] Shang, H., Zhang, Y., Lin, X., Yu, J.X.: Taming verification hardness: An efficient algorithm for testing subgraph isomorphism. In: VLDB. (2008) 364–375
- [Chen et al., VLDB'07] Chen, C., Yan, X., Yu, P.S., Han, J., Zhang, D.Q., Gu, X.: Towards graph containment search and indexing. In: VLDB. (2007) 926–937
- [Zhang et al., EDBT'09] Zhang, S., Li, J., Gao, H., Zou, Z.: A novel approach for efficient supergraph query processing on graph databases. In: EDBT. (2009) 204–215
- [Raymond et al., CJ'02] Raymond, J.W., Gardiner, E.J., Willett, P.: RASCAL: calculation of graph similarity using maximum common edge subgraphs. Comput. J. 45(6) (2002) 631–644
- [Yan et al., SIGMOD'05] Yan, X., Yu, P.S., Han, J.: Substructure similarity search in graph databases. In: SIGMOD Conference. (2005) 766–777

References

- [Shang et al., ICDE’10] Shang, H., Zhu, K., Lin, X., Zhang, Y., Ichise, R.: Similarity Search on Supergraph Containment . In: ICDE. (2010)
- [Ke et al., KDD’07] Ke, Y., Cheng, J., Ng, W.: Correlation search in graph databases. In: KDD. (2007) 390–399
- [Ke et al., SDM’09] Ke, Y., Cheng, J., Yu, J.X.: Top-k correlative graph mining. In: SDM. (2009) 1038–1049
- [Ke et al. ICDM’09] Ke, Y., Cheng, J., Yu, J.X.: Efficient discovery of frequent correlated subgraph pairs. In: ICDM. (2009) 239–248
- [Faloutsos and Tong, ICDE’09] Faloutsos, C., Tong, H.: Large graph mining: patterns, tools and case studies. In: ICDE (2009) tutorial

Thank you!

Q&A

香港中文大學
The Chinese University of Hong Kong

NANYANG
TECHNOLOGICAL
UNIVERSITY