



University of Illinois at Urbana-Champaign

Mining, Indexing & Query Processing on Graph Databases

Beespace Seminar Series

Peixiang Zhao

November 28th, 2008

Synopsis

- Introduction
 - Definition
 - Two Scenarios
- Graph Mining
- Graph Indexing and Query Processing
- Conclusion & Future Plans

Introduction

- Graph is a mathematical construct and a general data structure representing relations among entities
- The emergence and dominance of graphs ask for effective graph data management and mining tools so that users can organize, access, and analyze graph data in a way one might have not yet imagined
 - **Structural Pattern Mining:** *What are the hidden structural patterns and how can we find them?*
 - **Graph Indexing and Query Processing:** *How can we index graphs and perform searching, either exactly or approximately, on graphs*

Introduction

- Graph $G = (V, E, \Sigma, l)$
 - V : Nonempty vertex set
 - $E \subseteq V \times V$: Edge set
 - Σ : Alphabet for vertex and edge labels
 - l : Labeling function $l: (V \cup E) \longrightarrow \Sigma$
- **Specialized Graphs**
 - $|\Sigma| = 1$: *unlabeled* graph; $|\Sigma| > 1$: concept hierarchy may be induced
 - Weighted graph: more generalized form
 - E : unordered pairs of vertices: *undirected* graph, simple graph v.s. multigraph
 - Dense & Sparse graph
 - *Path*: (v_1, v_2, \dots, v_n) with length $(n-1)$, where $v_i \neq v_j$
 - *Connected* or *disconnected* graph
 - *Tree*: Connected, acyclic graph, *a.k.a. free tree*
 - Rooted Tree or Unrooted Tree
 - Ordered Tree or Unordered Tree

Two Examples



3D Hyperbolic Graphs of Internet Topology



Protein-Protein Interaction Network

Introduction

- Graph Representation

- Adjacency Matrix
- Adjacency List

- Common operations *w.r.t.* Mining & Indexing & Searching

- Entity & Relationship comparison: $O(1)$
- Neighbor finding: $O(V)$
- BFS & DFS: $O(V + E)$
- Reach-ability testing: $O(V^3)$
- (Sub-)Isomorphism testing: (Similar) Matching

	Path	Tree	Graph
Isomorphism	$O(n)$	$O(n)$	P or NPC (?)
Sub-Isomorphism	$O(n + m)$	$O(m^{3/2}n/\log m)$	NPC

- Graph Edit Distance (NPC)

Introduction

- Two scenarios
 - Graph Database
 - $G = \{g_1, g_2, \dots, g_N\}$
 - A Massive Network
 - Only one graph is considered
- Problem definitions, algorithms & solutions, and experimental settings are drastically different for two scenarios

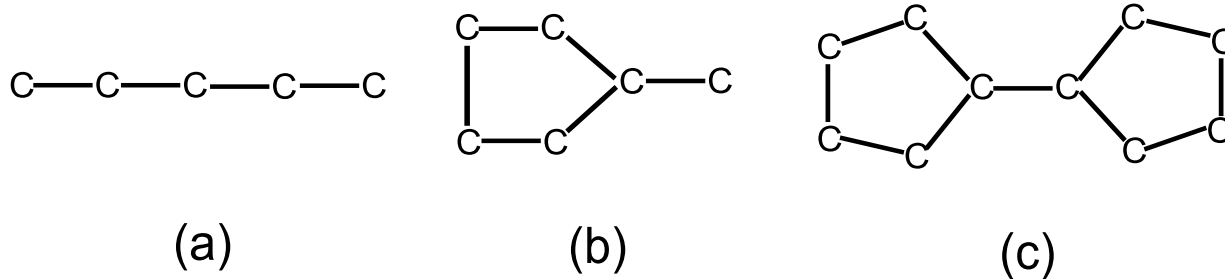
Graph Mining

- Mining Frequent Structural Patterns 😊
 - What?
 - Path, Tree, Graph
 - Others pertaining to special applications
 - Complete Set or Compressed One
 - Closed Frequent Patterns
 - Maximal Frequent patterns
 - Etc.
- Mining Outliers ☹️
 - What?

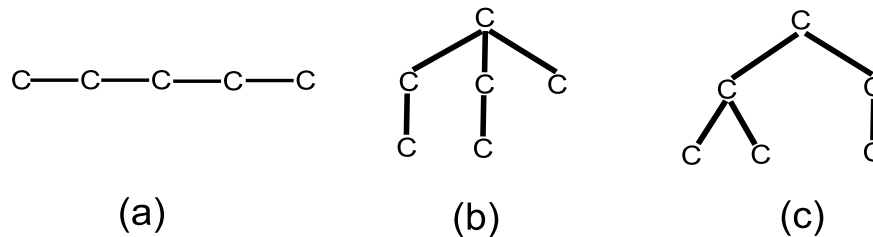
Graph Mining

- The Graph Database Scenario 😊
 - Given a graph database $G = \{g_1, g_2, \dots, g_N\}$, find frequent patterns p where p is *frequent* iff the ratio of graphs in G , that has p as its sub-pattern, is greater than or equal to a user-given threshold φ
 - Two Key Steps
 - Candidate generation
 - Frequency counting
- The Massive Network Scenario ☹️
 - How to define frequency of a structural pattern?
 - # (edge-disjoint) embeddings
 - Frequency Anti-monotone property may not hold

Mining Over Graph Database ($\phi = 2/3$)



A Sample Graph Database



Some Frequent Trees

Mining Over Massive Network

Freq(A) = 2

Freq(AB) = 2

Freq(ABC) = 2 ? 4 ?

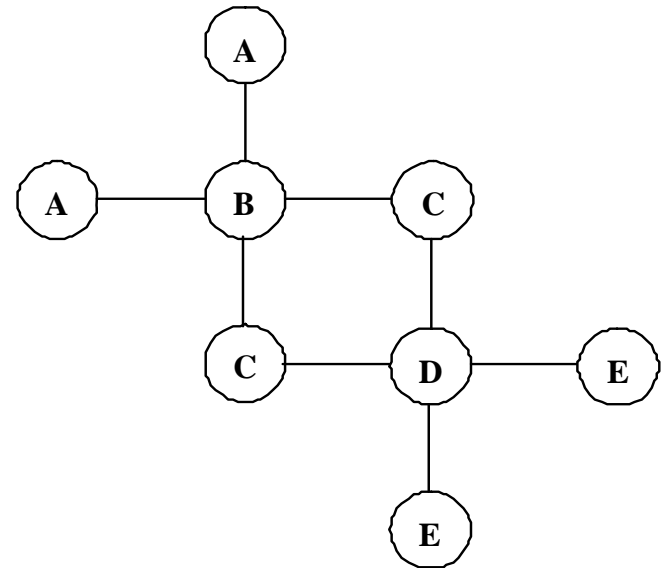
- **Computer scientists prefer 2, while**
- **Biologists prefer 4**

Freq(B) = 1

Freq(BC) = 2 !

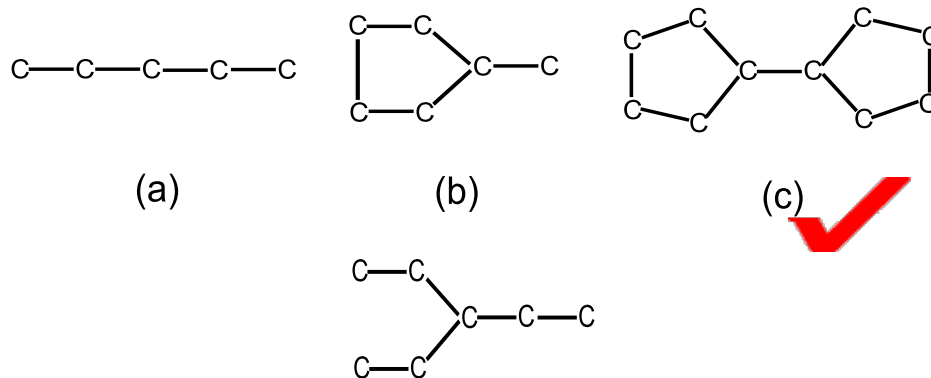
... ..

Freq(ABCDE) = 8 ? 2?



Graph Indexing & Query Processing

- Given a graph database $G = \{g_1, g_2, \dots, g_N\}$ and a query graph q , find the answer set $sup(q) = \{g_i \mid q \subseteq g_i, g_i \in G\}$
- NP-Hard, since subgraph-isomorphism checking is NPC
- Infeasible to check subgraph isomorphism sequentially for every g_i in G , especially challenging when graphs in G are large, or G is large and diverse
- Graph indexing!



Graph Indexing: Algorithmic Framework

- **Index construction** generates the index feature set F from the graph database G . For each feature f , $sup(f)$ is maintained
- **Query processing** is performed in a *filtering-verification* fashion:

- The filtering phase uses indexing features contained in q to compute the *candidate answer set*

$$C_q = \bigcap_{f \subseteq q \wedge f \in F} sup(f)$$

- Every graph in C_q contains all q 's indexed features. Therefore,
- The query answer set, $sup(q)$, is a subset of C_q
- The verification phase checks subgraph isomorphism for every graph in C_q . False positives are pruned and the real answer set $sup(q)$ is returned

Query Cost Model

- The cost of processing a graph query q upon G is modeled as

$$C = C_f + |C_q| \times C_v$$

- C_f : the filtering cost, and
- C_v : the verification cost (NP-Complete)
- **Several Facts:**
 - To improve query performance is to minimize $|C_q|$
 - The feature set F selected has great impacts on C_f and $|C_q|$
 - There is also an index construction cost, which is the cost of discovering the feature set F

Indexability of Path, Tree and Graph

- Frequent features (paths, trees, graphs) expose intrinsic characteristics of a graph database, G .
- Representatives to discriminate between different groups of graphs in a graph database
- Which one should we index? Path, Tree or Graph?
 - The frequent feature set size: $|F|$
 - The feature selection cost: C_{fs}
 - the candidate answer set size: $|C_q|$

Indexability of Path, Tree and Graph

- It is feasible and effective to select F_T or F_G , as indexing features for the graph query problem

	$ \mathcal{F} $	C_{FS}	$ C_q $	Comments
\mathcal{F}_P	small	low	large	simple structure, easy to be discovered, limited indexing ability
\mathcal{F}_T	large	intermediate	small	proper structure, easy to be discovered, good indexing ability
\mathcal{F}_G	large	high	small	complex structure, hard to be discovered, good indexing ability

Conclusion & Future Plans

- Which graph models should we apply in our Beespace project?
- Which scenario(s) should we choose? Either or both?
- What are of special interest to biologists, meanwhile, feasible in computer science?
 - Input
 - Σ : too large for mining meaningful structural patterns
 - $|\Sigma| = 1$: combinatorial explosion
 - Entities & Relations Graph
 - (Interactive) Mining language
 - Output ?

Conclusion & Future Plans

- What primitives are beneficial to Beespace and need to be supported?
 - Entities & Relation Identification (Generalization)
 - Find all C-H edges
 - Neighborhood retrieval
 - Who interact with a certain protein, and how?
 - Reachability query
 - Is protein A corelated with protein B, and how? (with a maximum of length of 5, say)
 - The shortest distance between A and B (weighted graph)
 - Frequent Structural Pattern (Path/Tree/Graph) Mining
 - Is the Benzene ring occurred more than 80 times in a graph database of 100?
 - Protein A is the largest graph and also frequent

Conclusion & Future Plans

- What primitives are beneficial to Beespace and need to be supported?
 - (Sub)Graph Matching
 - Protein A is the same as protein B (in an isomorphism sense)
 - Protein A subsumes protein B
 - Protein A is similar to protein B
 - What is the common part of protein A and protein B (maximum common subgraph)
 - Etc.



Thank you !

Q & A

