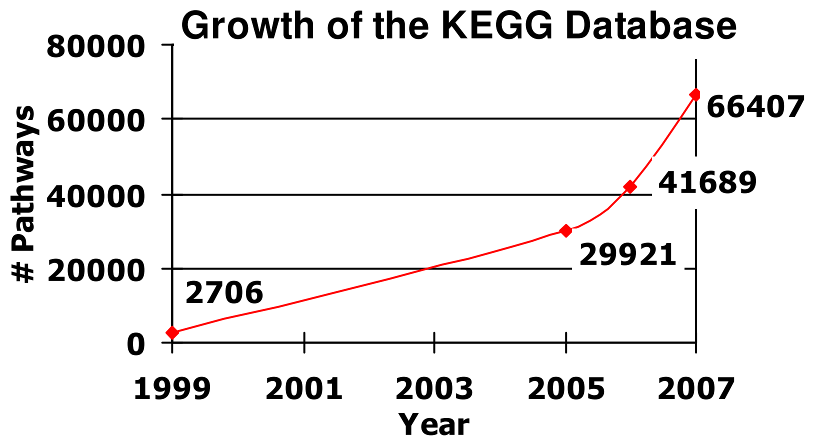
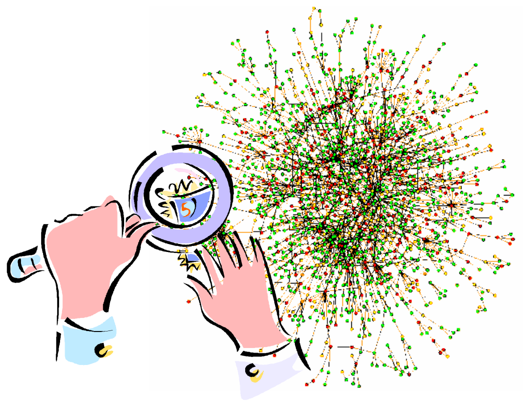
TALE: A Tool for Approximate Large Graph Matching

Yuanyuan Tian and Jignesh M. Patel
University of Michigan

Motivation

- Graphs are everywhere.
  - Social networks, computer networks, biological networks
- Graph databases are large and growing rapidly in size.



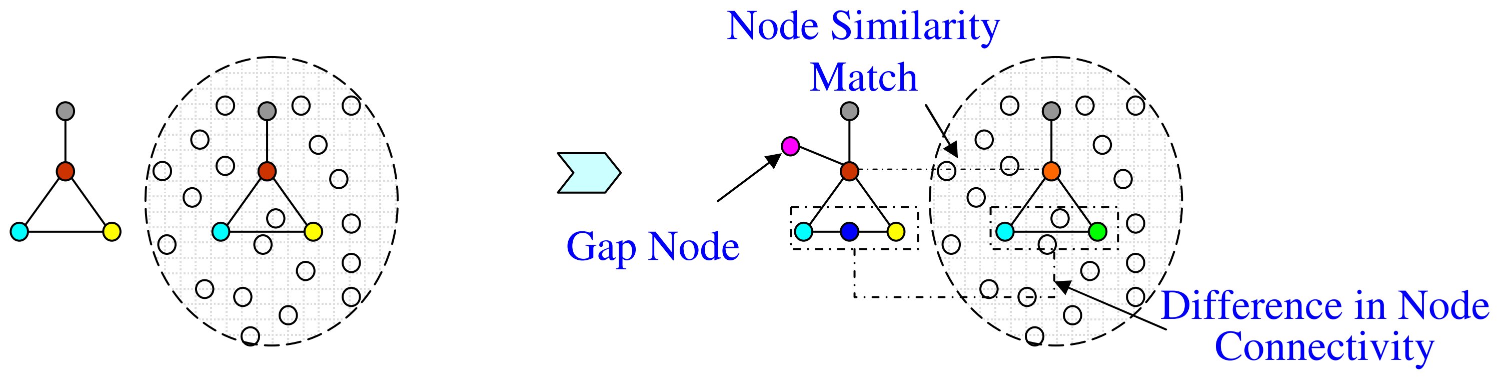


- Wealth of information is encoded in graph databases.

Need: Graph Matching

Motivation

- Previous studies largely focus on ***exact*** graph matching.
  - Assume precise graph data
  - Subgraph isomorphism (NP-Complete)
- Real life graphs are noisy and incomplete.
  - More challenging (need heuristic methods)



Need: Approximate Graph Matching

Motivation

- Most existing methods are applicable to small query graphs.
  - 10s of nodes and edges
- Supporting large queries is more and more desired.
  - Protein Interaction Networks (PINs):
    - 100s ~ 1000s nodes and edges
    - Compare PIN of one species against other species

Need: Approximate Large Graph Matching

TALE: A Tool for Approximate Large Graph Matching

- A Novel Disk-based Indexing Method
  - High pruning power
  - Linear index size with the database size
- Index-based Matching Algorithm
  - Significantly outperforms existing methods
  - Gracefully handles large queries and databases
- Experiments on Real Datasets
  - Effectiveness
  - Efficiency

TALE: A Tool for Approximate Large Graph Matching

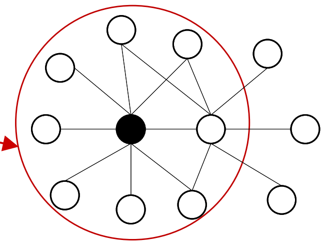
- **A Novel Disk-based Indexing Method**
  - High pruning power
  - Linear index size with the database size
- Index-based Matching Algorithm
  - Significantly outperforms existing methods
  - Gracefully handles large queries and databases
- Experiments on Real Datasets
  - Effectiveness
  - Efficiency

Neighborhood Indexing

- Index Unit?

Neighborhood

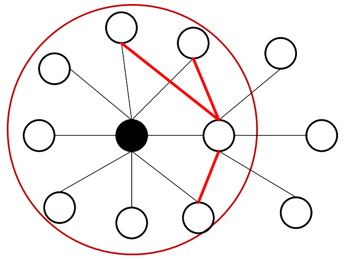
(induced subgraph of a node and its neighbors)



Index Unit	Pruning Power	Index Size
Subgraphs	High ☺	$O(n^k)$ ☹
Nodes	Low ☹	$O(n)$ ☺
Neighborhoods	High ☺	$O(n)$ ☺

Index Unit

- Index Unit: Neighborhood
  - Which node is at the center?
    - Node label
  - How many neighbors does the node have?
    - Node degree
  - How do the neighbors connect to each other?
    - NeighborConnection: # edges between neighbors
  - Who are the neighbors?



Index Unit

- Who are the neighbors?
  - Naïve approach: list the labels of the neighbors
    - Problem: the number of neighbors varies.
  - If # labels in the problem domain is a small constant.
    - Deterministic bit array.

A	B	C	D	E
1	0	0	1	1

$\longrightarrow$ Neighbor Array

  - What if the number of labels is huge?
    - Bloom filter: label $\xrightarrow{\text{hash}}$ position in a m-bit array.
- Information in the index unit
  - (label, degree, nConn, nArray)

Match a Query Neighborhood

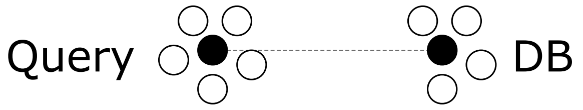
Exact

- ✓ N_q.label = N_{db}.label
- ✓ N_q.degree ≤ N_{db}.degree
- ✓ N_q.nConn ≤ N_{db}.nConn
- ✓ (**NOT** N_{db}.nArray) **AND** N_q.nArray = 0

Approximate

group nodes based on similarity

- ✓ *group*(N_q.label) = *group*(N_{db}.label)
- ✓ N_q.degree ≤ N_{db}.degree + ε
- ✓ N_q.nConn ≤ N_{db}.nConn + δ
- ✓ |(**NOT** N_{db}.nArray) **AND** N_q.nArray| ≤ ε



ρ : % of neighbors of a query node with no corresponding matches in the neighborhood of a database node

max # missing neighbors: $\varepsilon = \rho\,(N_q.\text{degree})$

max # missing nConn: $\delta = \varepsilon(\varepsilon - 1)/2 + \varepsilon(N_q.\text{degree} - \varepsilon)$

Index Structure

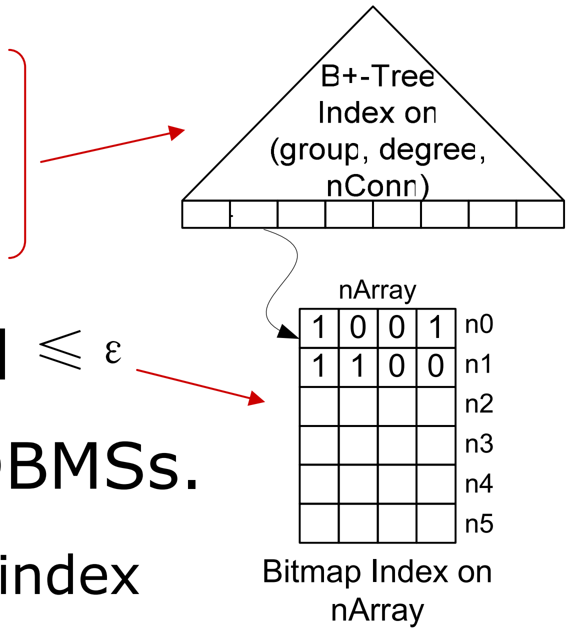
- Support efficient search for DB neighborhoods.

  $\mathit{group}(N_{db}.\text{label}) = \mathit{group}(N_q.\text{label})$

  $N_{db}.\text{degree} \geqslant N_q.\text{degree} - \varepsilon$

  $N_{db}.\text{nConn} \geqslant N_q.\text{nConn} - \delta$

  $|(\textbf{NOT}\ N_{db}.\text{nArray})\ \textbf{AND}\ N_q.\text{nArray}| \leqslant \varepsilon$

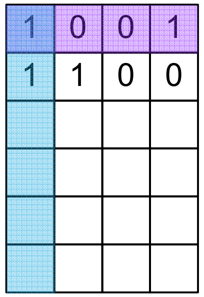


- Simple implementation in RDBMSs.
  - Use existing robust disk-based index structures in RDBMSs.

Hybrid Index Structure

Index Probing

- Probe the B+tree for group, degree and nConn
  - Easy
- Probe bitmaps for nArrays
  - Naïve approach: look at each row of a bitmap
  - A better approach
    - Operate on bit slices.
    - Up to 12X speedup!



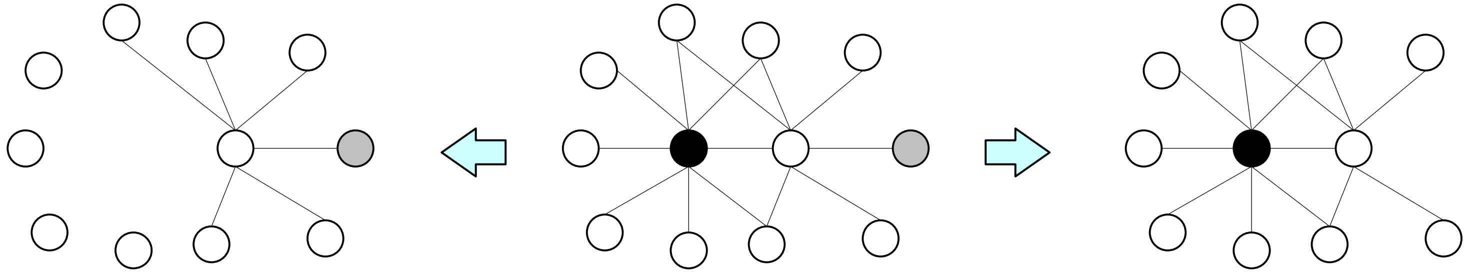
1	0	0	1
1	1	0	0

TALE: A Tool for Approximate Large Graph Matching

- A Novel Disk-based Indexing Method
  - High pruning power
  - Linear index size with the database size
- **Index-based Matching Algorithm**
  - Significantly outperforms existing methods
  - Gracefully handles large queries and databases
- Experiments on Real Datasets
  - Effectiveness
  - Efficiency

Observations

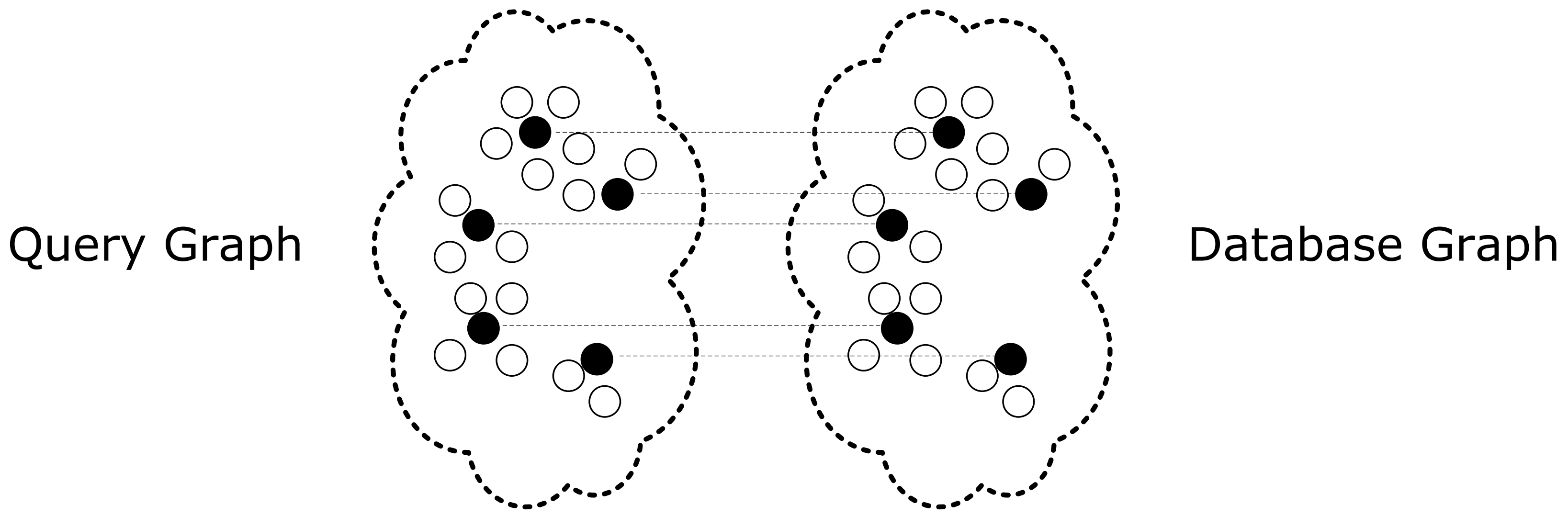
- Observation 1: Not every node plays the same role in a graph.
  - Node importance



- Observation 2: A good match should be more tolerant towards missing unimportant nodes than missing important nodes.

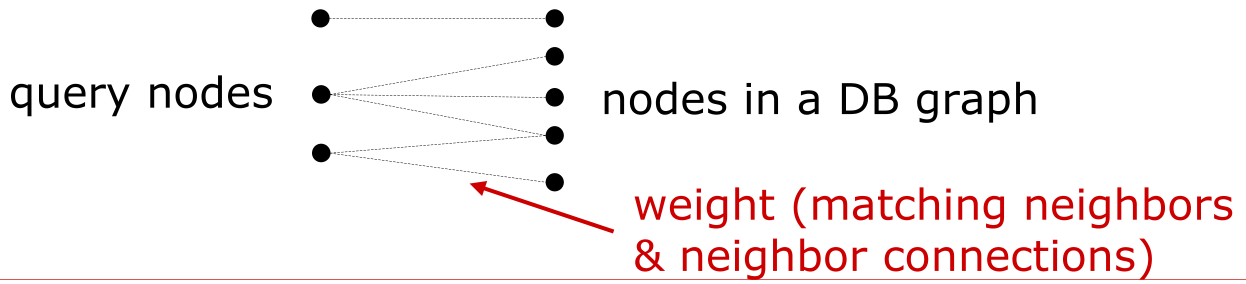
Matching Algorithm Overview

- Step 1: Match the important nodes from the query.
- Step 2: Progressively extends the node matches.



TALE Matching Algorithm

- Step 1: Match important nodes from the query.
  - Select important nodes.
    - Importance measure: degree centrality
    - The percentage of important nodes: P
  - Probe Neighborhood Index to match important nodes.
  - For each candidate graph in the database, find the one-to-one mappings to the important query nodes.
    - Maximum weighted bipartite graph matching



TALE Matching Algorithm

- Step 2: Progressively extends the node matches.
  - Start from the importance node matches.
  - Match “nearby” nodes of already matched nodes.
    - Not just immediate neighbors
    - Also nodes two hops away

      → gap nodes

      → differences in node connectivity

TALE: A Tool for Approximate Large Graph Matching

- A Novel Disk-based Indexing Method
  - High pruning power
  - Linear index size with the database size
- Index-based Matching Algorithm
  - Significantly outperforms existing methods
  - Gracefully handles large queries and databases
- **Experiments on Real Datasets**
  - Effectiveness
  - Efficiency

Experimental Evaluation

- Implementation
  - C++ on top of PostgreSQL
- Evaluation Platform
  - 2.8GHz P4, 2GB RAM, 250GB SATA disk, FC2
  - PostgreSQL: version 8.1.3, 512 MB buffer pool
- Experimental Datasets
  - BIND protein interaction networks
  - ASTRAL protein structures
- Evaluation Measures:
  - Effectiveness
  - Efficiency

Effectiveness Experiment

- Protein Interaction Network Comparison (BIND)

	#node	#edge
rat	830	942
mouse	2991	3347
human	8470	11260

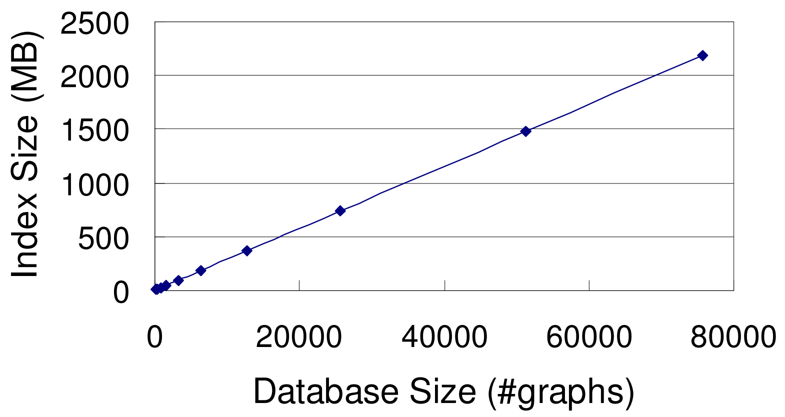
	#KEGGs hit	KEGG coverage	Time (sec)
rat vs. human			
Graemlin	0	NA	910.0
TALE	6	3.2%	0.3
mouse vs. human			
Graemlin	18	5.0%	16305.5
TALE	42	13.6%	0.8

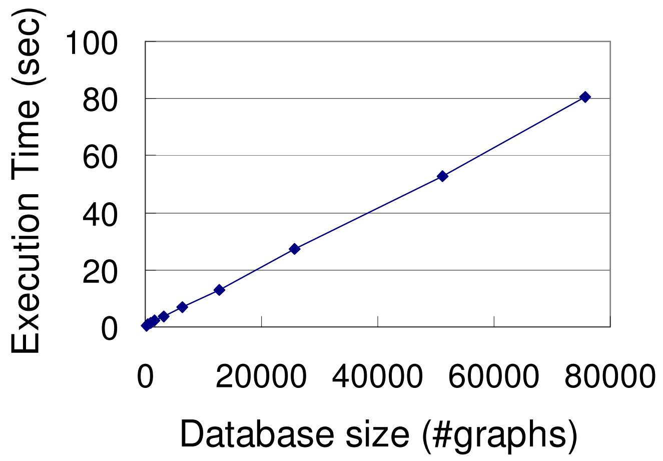
# KEGGs hit: number of pathways aligned between 2 species

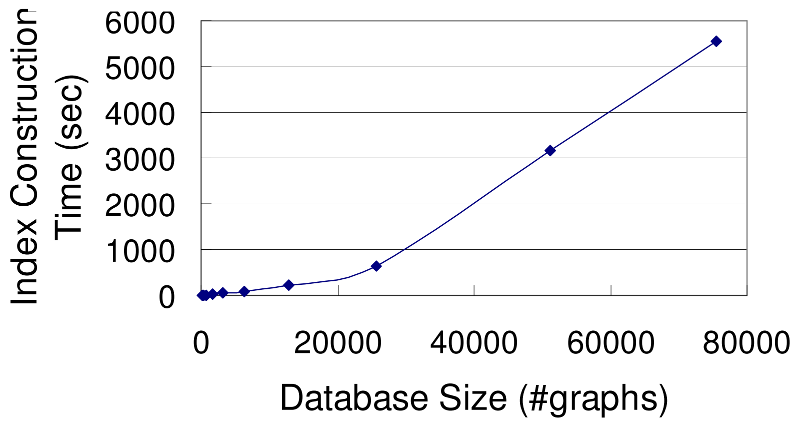
KEGG coverage: fraction of proteins aligned within a pathway.

Efficiency Experiment

- Query increasing sized ASTRAL datasets
  - 20 queries (153.1n, 592.0e)
  - Top 20 results







Related Work

- Index-based Approximate Graph Matching
  - Graphfil, PIS, CDIndex, C-Tree, SAGA
  - Limited approximation: Graphfil, PIS, CDIndex, C-Tree
  - For small queries: Graphfil, PIS, CDIndex, SAGA
- Pairwise Graph Alignment Methods
  - NetworkBlast, MaWIsh, Graemlin
  - Specific to protein interaction networks
  - Very slow for database search (no index)

Conclusion

- TALE → Approximate Large Graph Matching
- Neighborhood Indexing
  - Disk-based index using existing index structures in RDBMSs
  - High pruning power
  - Linear index size with the database size
- Index-based Matching Algorithm
  - Distinguish nodes by importance
  - Match important nodes then extend to others
- Experiments on Real Datasets
  - Improved effectiveness and efficiency over existing methods

Questions?

Suggestions?

Thanks! ☺