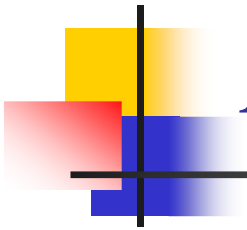
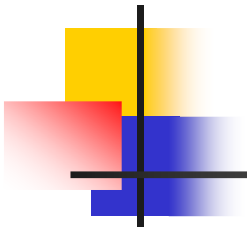
A FLEXIBLE GRAPH PATTERN MATCHING FRAMEWORK VIA INDEXING

Wei Jin and Jiong Yang
EECS Department
Case Western Reserve University



Agenda

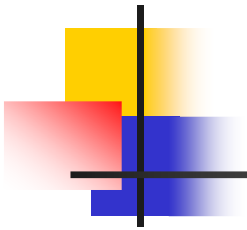
1. Introduction and problem definitions
2. Indexing methods
2.1 Global Index
2.2 Local Index
3. Matching Algorithm
3.1 Retrieve Complete Set of Matches
3.2 Top-k Matches Scheme
4. Experimental Results



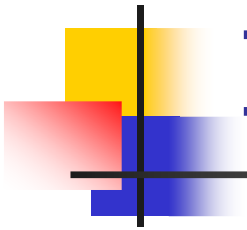
Introduction

- Many data is in the form of graph (PPIN, Facebook, map, software dependency graph)
- Graph pattern matching query
- Previous work only allows one uniform relationship on all edges of the query graph pattern.
- We propose a flexible query model

A Flexible Pattern Matching Framework



- The edges in the query graph represent relationships specified by an interval $[min_e, max_e]$
- The shortest distance between two matched vertices in the database graph should fall into the corresponding interval.
- The first attempt to model different relationships in a single pattern, which previous models cannot represent.



Introduction

- A straightforward solution - Naïve Join

1. The shortest distance calculation (Dijkstra) is costly: O(| E | + | V | log | V |)
2. To join m sets, each of which is of length n, the complexity is O(n^m)

- Index is used to estimate the shortest distance; shrink the candidate sets.

Preliminaries

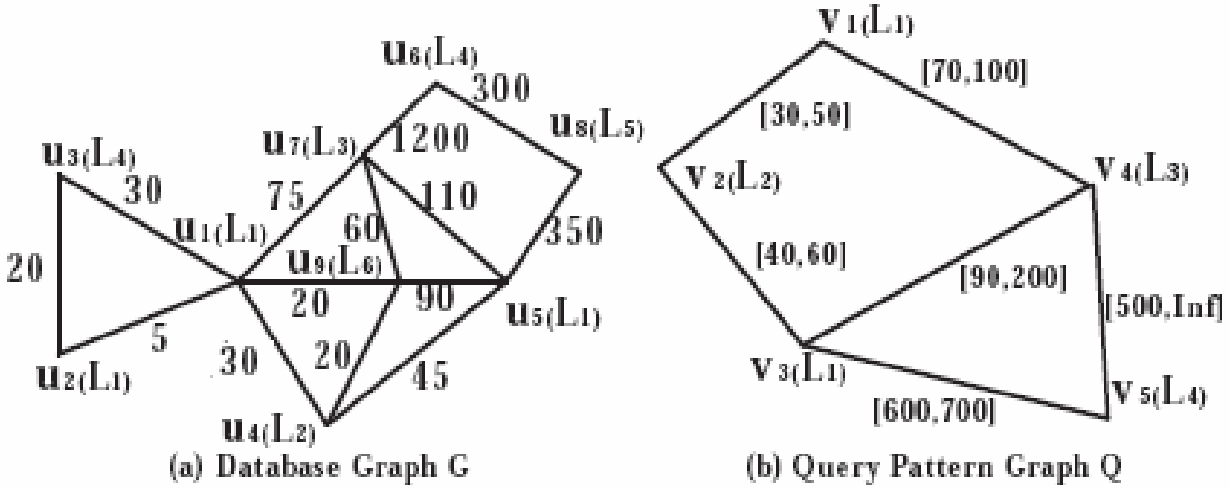
Definition 1 *A **labeled database graph** G is a five element tuple* $G = \{V_G, E_G, \Sigma_{V_G}, L_V, W_E\}$

Definition 2 *The **query pattern graph*** $Q =< V_Q, E_Q, L_V, \Delta >$

Definition 3 *a **match** of Q in G is a set of y distinctive vertices $u_1, u_2, ..., u_y$ in G*

- Problem Statement: Given a large database graph G, and a query graph Q, we want to efficiently find all the matches of Q in G with the help of the indexed information.

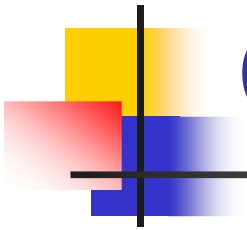
An example of the problem



(a) Database Graph G (b) Query Pattern Graph Q

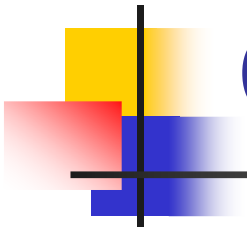
The Problem and Matches

For query Q with vertices $(v_1, v_2, v_3, v_4, v_5)$, there are two matches of Q: $(u_1, u_4, u_5, u_7, u_6)$ and $(u_2, u_4, u_5, u_7, u_6)$



Global Index

- Building algorithm: A variation of K-medoids
- K seeds are randomly selected
- A is the average number of clusters one vertex belongs to
- Each seed is grown to a cluster until |V(G)|*A/K vertices are visited.
- Assign the unvisited vertices to A closet seeds.



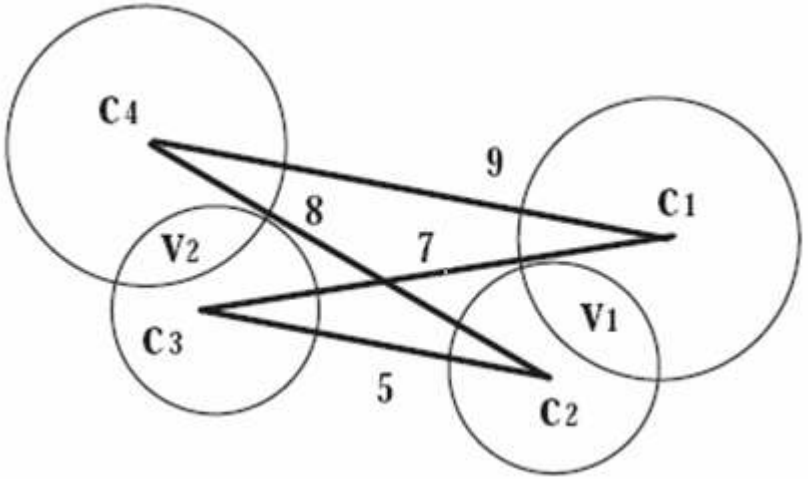
Global Index

- Ci: the center of a cluster
- d(v1, v2): the shortest distance between vertex v1 and v2
- Dxy = d(Cx, Cy): the highway distance
- Suppose vi, vj belong to Cx, Cy

[Dxy-d(x,vi)-d(y,vj), Dxy+d(x,vi)+d(y,vj)]

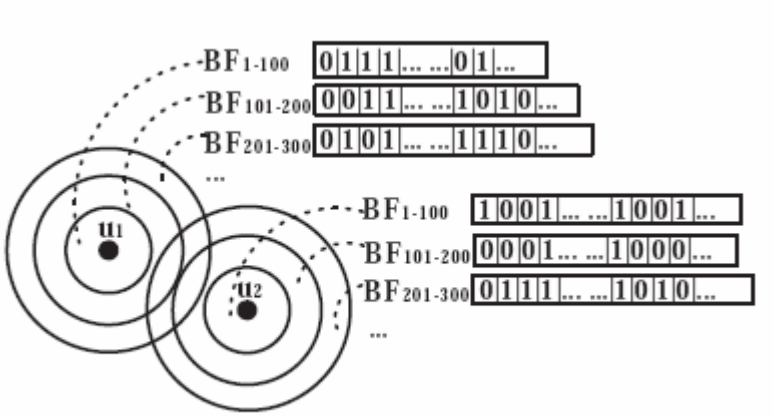
An Example of the Global Index

- Suppose d(c1, v1) = 2, d(c2, v1) = 2, d(c4, v2) =3 and d(c3, v2) = 2
- C1,C4: [9-2-3,9+2+3]=[4,14]
- C1,C3: [3,11]
- C2,C4: [3,13]
- C2,C3: [1,9]
- Final: [4, 9]

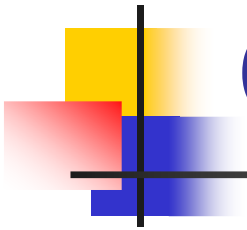


Bloom Filter Based Local Index

- The neighborhood of a vertex up to Max_dist distance is break into rings, each of which corresponds to a bloom filter.



The General Structure of the Indices



Query Algorithm

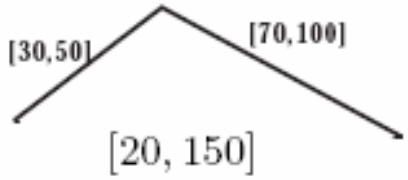
- 1.Processing the query graph to obtain implicit relationships among vertices;
- 2. Generating candidate vertex matches by the label index of the local index;
- 3. Generating candidate pattern matches via the vertex index of the local index and the global index;
- 4. Verifying the candidate matches to eliminate false positives

Query Preprocessing

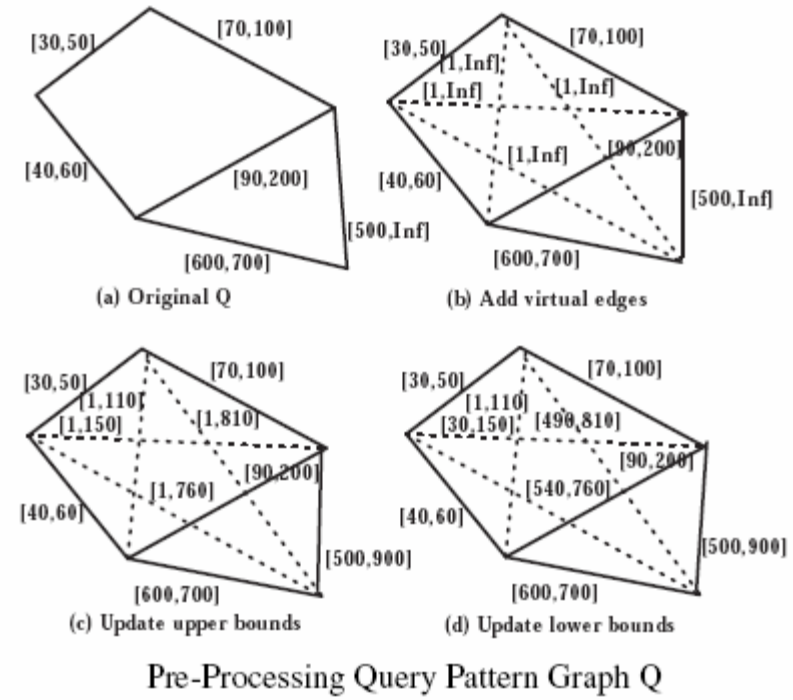
- Infer implicit relationship from explicit relationship specified by the user through the triangle equality.

$r_{1,2} = [min_1, max_1]$ and $r_{2,3} = [min_2, max_2]$

the range $r_{1,3}$ will be $[min_2 - max_1\ ,\ max_1 + max_2]$

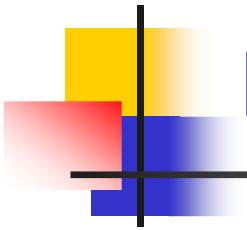


An example



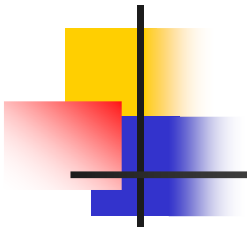
Pre-Processing Query Pattern Graph Q

Generating candidate vertex matches by the label index

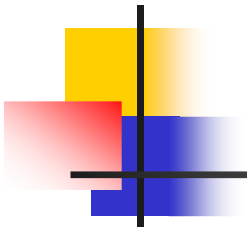


- For each vertex v_i in Q, a candidate set CL_i is constructed, containing all vertices in G that share the same label as v_i
- Prune these CL sets by label vertex

Generating Candidate Pattern Matches



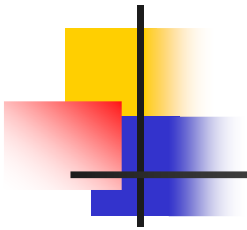
- The match discovery process can be considered as a repeated join process.
- It is very expensive if directly calculating the shortest distance on G.
- Perform join by indices to quickly prune unqualified matches. Place links on qualified matches.
- Travel on the graph with links to get final matches



Match Verification

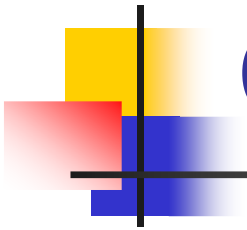
- Verify the candidate matches

1) The bloom filter data structure inherently has a small false positive rate
2) The range on an edge could not be exactly covered by one or multiple indexed rings
3) The estimation interval produced by the global index overlaps with the edge interval



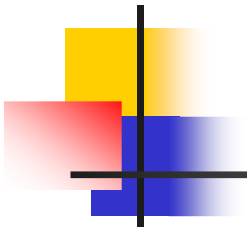
Match Verification

- Involves calculating the shortest distance on G.
- Use 2-hop labeling technique.
- The space is $O(|V(G)| * |E(G)|^{1/2})$ and time complexity is $O(|E(G)|^{1/2})$



Generating Top-k Matches

- Use the sum of the shortest distance as the score of a match. Smaller score means a more compact match.
- Two priority queues are maintained, one for k complete matches, one for partial matches
- First generate k complete matches by a depth first search, obtain the pruning threshold
- Expand partial matches by a breadth first search. Update the threshold if necessary.
- Stop until the score of the current partial match exceeds the threshold



Experimental Results

- The flexible graph pattern framework (FGP) is compared with D-Join, the most related work, on a set of real and synthetic data sets
- Several versions of FGP are tested: FGP, FGP-GLOBAL, FGP-BOTH, FGP-TOPK

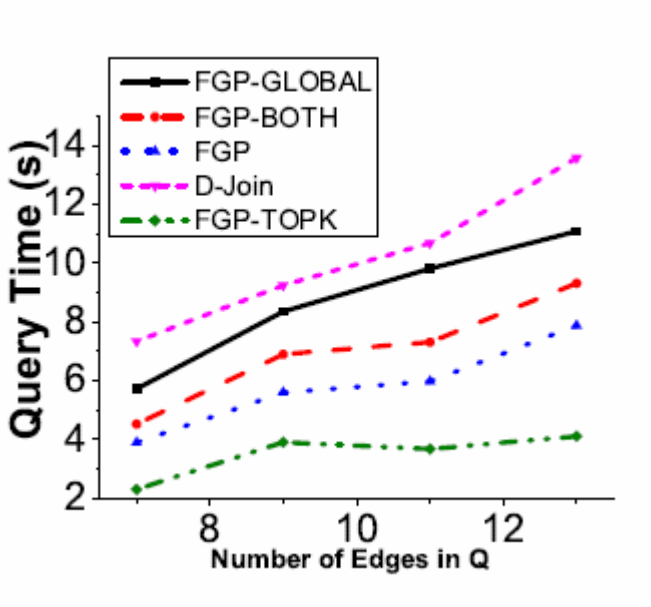
Experiments on Real Data Sets

- A protein interaction network for homo sapiens is used here.
- 6410 vertices, 27005 edges, and 632 distinct labels. The average degree is 8.4, $K = 600$, $A = 3$
- Index building

FGP: 174s, 32MB

D-Join: 238s, 5.1MB

- Four known signal transduction pathways from the KEGG database are used as the query pattern graph Q.



Experiments on Real Data Sets

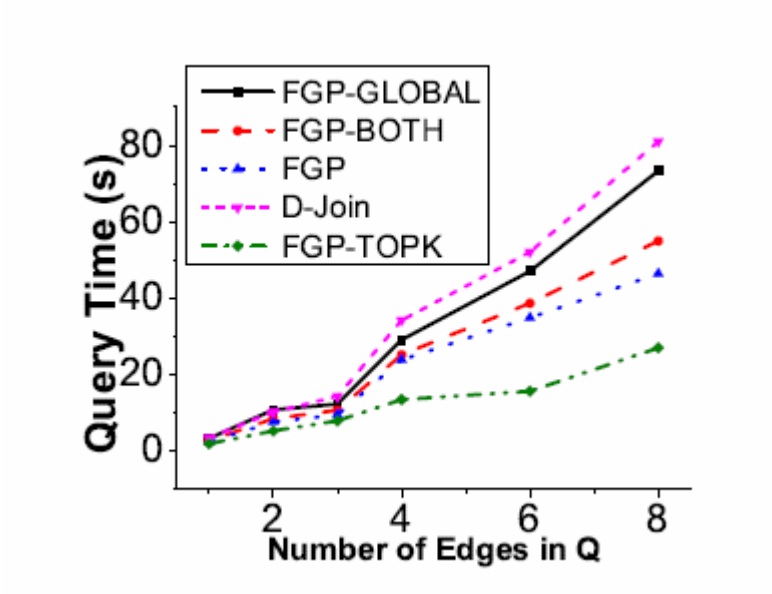
- Second data set: a portion of the Facebook users network.
- 21361 vertices, 320244 edges, 1000 labels, average degree: 30, $K = 1500$, $A = 4$
- Index building:

FGP: 86min, 234MB

D-Join: 142min, 21MB

- six query graphs

have 2, 3, 3, 4, 4, 5 vertices and 1, 2, 3, 4, 6, 8 edges



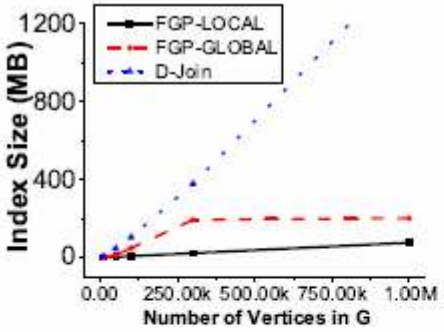
Experiments on Synthetic Data Sets

- We use five parameters to analyze the performance. Each time we vary one parameter and others remain default values.

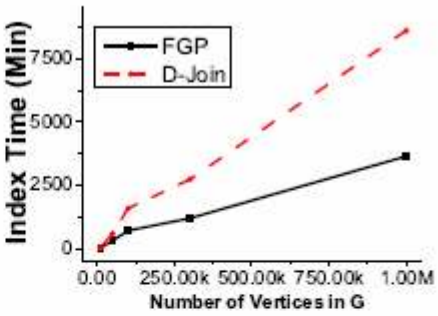
Table 4.1: Default Parameter Value

Parameter	Default Value
Number of vertices in G	50K
Average degree of G	7.5
Max_Dist	1000
Num_Interv	10
Number of edges in Q	4

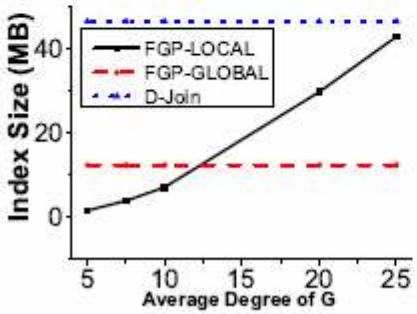
Index Size and Buiding Time



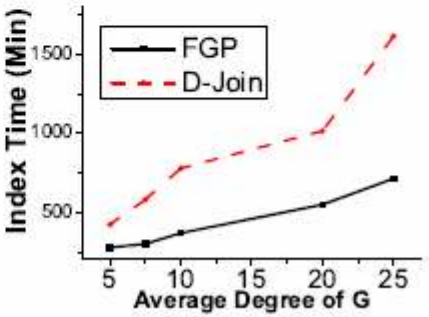
(a) Various $|V(G)|$



(b) Various $|V(G)|$

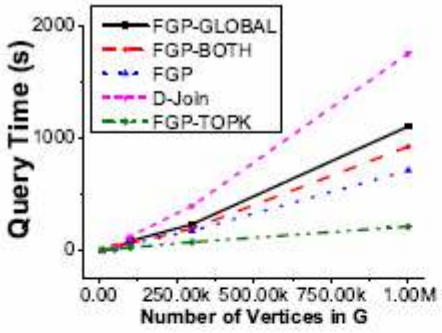


(c) Various deg(G)

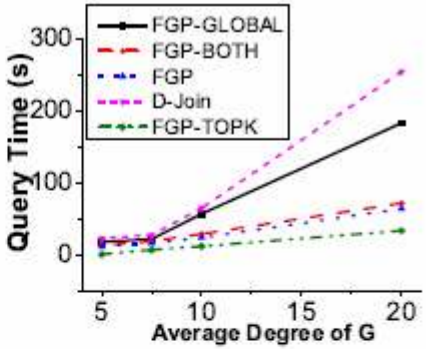


(d) Various deg(G)

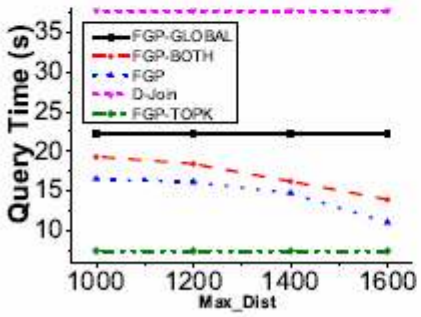
Query Time



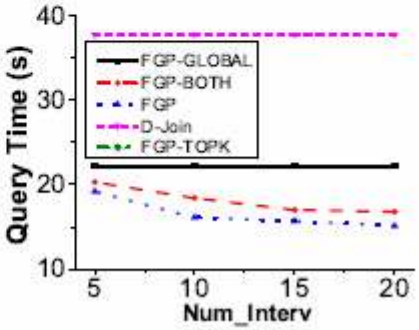
(a) Various $|V(G)|$



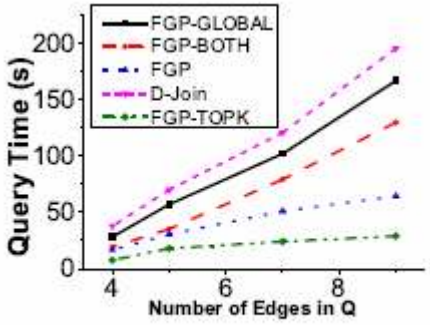
(b) Various deg(G)



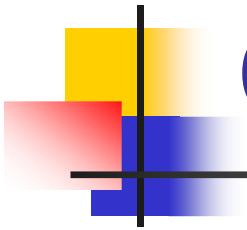
(c) Various Max_Dist



(d) Various Num_Interv

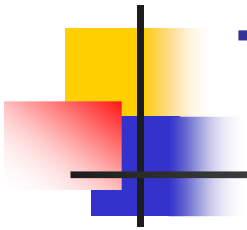


(e) Various $|E(Q)|$



Conclusions

- A flexible pattern matching model is first proposed;
- Two types of index, local and global index are constructed;
- A novel matching algorithm is devised to efficiently retrieve the matches;
- A top-k matching scheme is also devised.



Thank You

Q&A