Advanced Data Management

Medha Atre

Office: KD-219
atrem@cse.iitk.ac.in

Oct 24, 2016

What we learnt until now

- Graph indexing using relational databases.
- Graph pattern query processing using relational joins.
- Various ways of indexing the graphs for the basic graph pattern queries – queries with wildcard nodes.
- Pruning methods using semi-joins and variants of that.
- Distribution of the graph data – taking care of the skew.
- Distributed basic graph pattern query processing.

Special topics

- Reachability queries – useful in social networks, intelligence agencies, biological pathways.
- Reachability queries – total and partial transitive closure methods. Trade-off between indexing construction time, index size, and query processing time.
- Regular path queries – Special case of reachability queries where a "regular expression" puts additional constraints on the *type* of reachability between nodes.
- Keyword search over relational DB – schema-based and schema-less approaches.
- Keyword search over graph DB – graph summarization, pruning the search space, top-k SPARQL queries and answers, r-cliques etc.

Other graph indexing methods

- Graph indexing as a whole has applications in Machine Learning, Image Segmentation and Analysis, and Data Mining topics like clustering.
- Abstract concept of "feature" – can be paths, subgraps, subtrees.
- Graph DB can consists of one large graph or several smaller graphs.
- In this course, we mainly studied the former, and now we will take an overview of the latter.

Type of queries

- **Substructure Search** – Given a graph database $D = \{G_1, G_2, ..., G_n\}$ and a query graph Q, substructure search is to find all the graphs that contain Q.
- Same as before, two main components of this query processing:
  1. Index construction.
  2. Query processing – Searching, Fetching (candidate results), Verification and pruning (of the candidate results to get the final results).

Indexing Paths

- Using "paths" as features, enumerate all paths in the graphs in the DB upto *maxL* length.
- Main applications in XML and path queries. For substructure queries, one needs to maintain additional information about adjacency so as to join the filtered candidate paths in order to get the query answers.

Indexing Substructures

- Different from the kind of indexing we saw in case of RDF graphs.
- Useful in chemical networks, recommender systems with graphical representation.
- **Frequent Structures** – Given a graph database $D = \{G_1, G_2, ..., G_n\}$ and a graph structure f, the support of f is defined as $sup(f) = D_f$, whereas D_f is referred as f's supporting graphs. With a predefined threshold min_sup, f is said to be frequent if $sup(f) \geqslant min\_sup$.

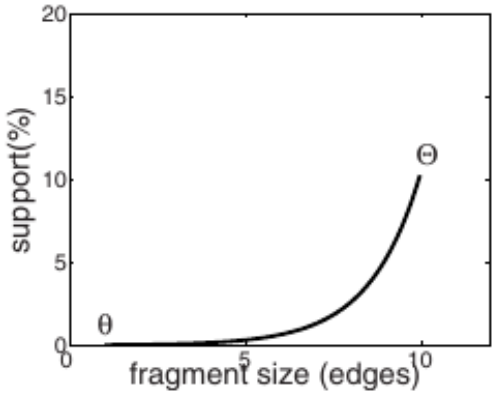
Indexing Substructures

- Given a query graph Q, if Q is frequent, the graphs containing Q can be retrieved directly (as Q is indexed).
- Else, sort all Q's subgraphs in the support decreasing order: $f_1, f_2, ..., f_n$. There must exist a boundary between f_i and f_{i+1} where $D_{f_i} \geqslant min\_sup$ and $D_{f_{i+1}} < min\_sup$.
- Since all the frequent structures with minimum support are indexed, one can compute the candidate answer set C_Q by $\bigcap_{1 \leq j \leq i} D_{f_j}$.

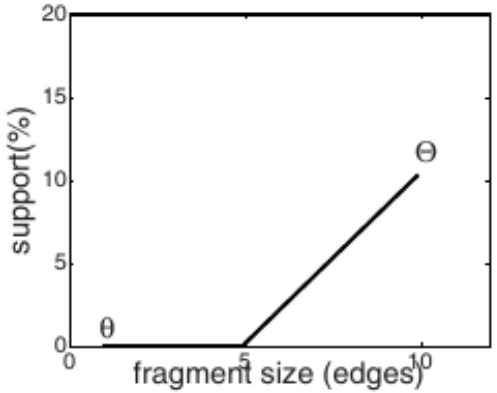
Ways of Indexing Substructures

- For low support queries the size of candidate answer set C_Q is related to the setting of *min_sup*.
- If *min_sup* too high, C_Q might be very large. If *min_sup* too low, difficult to generate all the frequent structures due to the exponential pattern space.
- Hence *Size Increasing Support Constraint* is used.
- **Size-increasing Support** – Given a monotonically nondecreasing function, $\tau(l)$, structure f is frequent under the size-increasing support constraint if and only if $D_f \geqslant \tau(size(f))$, and $\tau(l)$ is a size-increasing support function.

Example Size-increasing Support Functions



(a) Exponential



(b) Piecewise-linear

Figure from "Graph Indexing" chapter in "Managing and Mining Graph Data", Springer 2010.

Discriminative Structures

- Often wise to index *smallest common substructures* as more query graphs may contain these structures.
- If f', is a supergraph of f and has the same support as f, it will not be able to provide more information than f if both are selected as indexing features.
- f' is not more *discriminative* than f.
- **Redundant Structure** – Structure x is redundant with respect to a feature set F if D_x is close to $\bigcap_{f \in F \wedge f \subseteq x} D_f$.
- x not used as an indexing feature as it does not provide any new benefits.

Discriminative Structures

- If $f_1, f_2, ..., f_n$ be the indexing structures (features). Given a new structure x, the discriminative power of x is measured by $Pr(x|f_{\varphi_1}, ..., f_{\varphi_m})$, $f_{\varphi_i} \subseteq x$, $1 \leqslant \varphi_i \leqslant n$.
- This is the probability of observing x in a graph given the presence of $f_{\varphi_1}, ..., f_{\varphi_m}$. Discriminative ratio, γ defined as $1/(Pr(xf_{\varphi_1}, ..., f_{\varphi_m})$, which could be calculated as:

$$\gamma = \frac{|\bigcap_i D_{f_{\varphi_i}}|}{|D_x|}$$

D_x is the set of graphs containing x and $\bigcap_i D_{f_{\varphi_i}}$ is the set of graphs containing the features belonging to x.