Advanced Data Management

Medha Atre

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

Oct 27, 2016

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.

Graph indexing methods

- **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.
- 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.

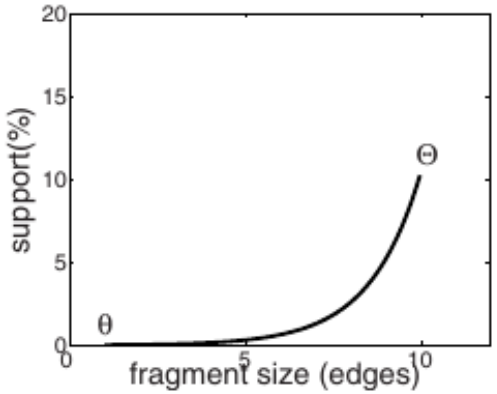
Indexing Substructures

- **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 $sup(f) \geqslant min\_sup$.
- 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 $\bigcap_{1 \leq j \leq i} D_{f_j}$.

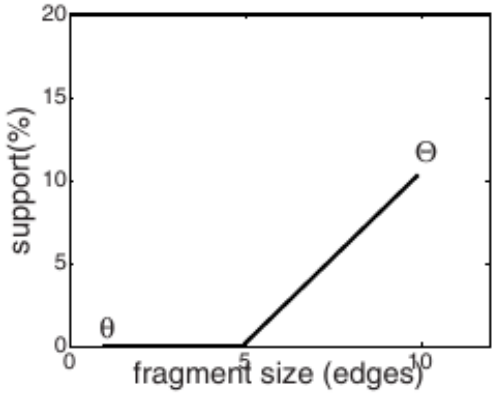
Ways of Indexing Substructures

- 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.

Closed Frequent Structures

- If the query graph is frequent (indexed as frequent substructure), answer set is available immediately.
- If it is not frequent, the candidate answer set obtained from the indexed frequent substructures is likely small, hence easier to verify and generate final answer set.
- δ-Tolerance Closed Frequent Subgraphs compress a set of frequent subgraphs, and each such compressed δ-TCFG is a representative *supergraph* of that set of frequent subgraphs.
- Build an inverted index on these δ-TCFGs and keep it in memory.
- Thus build a two-level index structure.

Tree+Δ

- Majority of the frequent graph patterns discovered in many applications are tree structures.
- Tree mining can be performed more efficiently than general subgraph structures.
- Tree+Δ first mines and indexes frequent trees.
- Selects a small number of *discriminative* graph structures from the query, at runtime.
- Pruning power of these discriminative structures is estimated by the subtree features contained within them.

Tree+Δ

- If Q is non-tree cyclic, discriminative substructures f are built into an inverted index on the fly, or used from previously executed queries. These are $D_f \subseteq D$.
- All frequent subtrees of Q upto maximum size $maxL$ are considered. These are $T(Q)$.
- From $T(Q)$ candidate answer set C_Q is built, considering the graphs indexed per $t \in T(Q)$.
- Do $C_Q \cap D_f$, and verify each graph in this intersection.

GString

- Combines three basic structures together: path, star, cycle.
- First extract all the cycles, then extract the stars and paths in the remaining DB.
- Essentially transforms graphs into "string" representations and treats substructure search as substring search.
- Uses suffix trees to perform indexing and search.

Hierarchical Indexing

- **Closure-tree**: graph organized hierarchically. Each node in the hierarchy constains summary info of about its descendants.
- The summary nodes in the hierarchy created by considering two graphs, isomorphism mapping between them, and then doing elementwise union of them.
- At query time, traverse this Closure tree, and prune nodes based on pseudo subgraph isomorphism, i.e., approximate subgraph isomorphism testing with high accuracy and low cost.
- Verify the candidate set.

Structure Similarity Search

- Used when the query has no match or very few matches in the graph DB.
- Given two graphs G and Q, if P is the maximum common subgraph of G and Q, then the *substructure similarity* between G and Q is defined by $\frac{E(P)}{E(Q)}$, and $\theta = 1 - \frac{E(P)}{E(Q)}$ is called the *relaxation ratio*.
- The indexed feature set is used to do the approximate matching of Q in the graph DB.