Enhancing Graph Database Indexing By Suffix Tree Structure

Bonnici V[(1)], Di Natale R[(1)], Ferro A[(1)], Giugno R[(1)], Mongiovi M[(1)], Pigola G[(1)], Pulvirenti A[(1)], Shasha D[(2)]

[(1)] Department Of Computer Science, University of catania, Catania
[(2)] Courant Institute of Mathematical Sciences, New York University, New York

Motivation

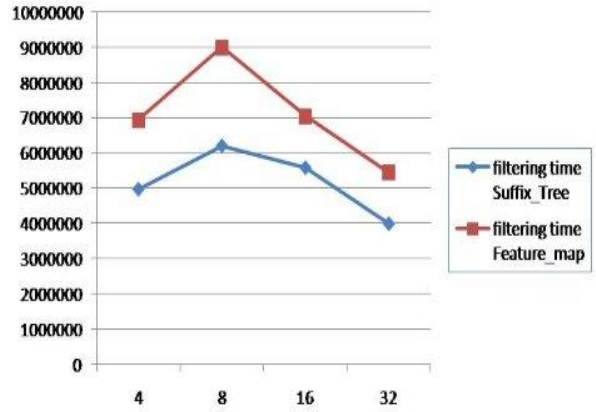
Many applications in industry, science, and engineering share the same problem: given a subgraph, find its occurrences in a database of graphs. The increasing size of databases application requires efficient structure searching algorithms. Examples of such databases and substructure searching methods can be found in computational chemistry and biology. For example, in drug discovery, the main task is to find novel bioactive molecules, i.e., chemical compounds that protect human cells against a virus. In computational biology querying for subgraphs, matching a specific topology, is useful to find motifs of networks that may have functional relevance. Finding occurrences of a subgraph in a set of graphs is known to be NP-Complete. Although graph-to-graph matching algorithms can be used, efficiency considerations suggest the use of specific techniques to reduce the search space and the time complexity. In a preprocessing phase, each graph of the database is analyzed in order to extract and store its discriminant features together with the number of their occurrences. Such structures constitute the graph database index. The features could be either all the paths up to a certain length (GraphGrep, GraphFind) or a set of suitable trees or subgraphs (GIndex, FgIndex, Tree+Delta) computed by mining techniques. For each indexed feature, all graphs containing it are maintained. All such informations are stored by using a hash table (GraphGrep) or by compacting features in a trie (GIndex). In the filtering phase, the graph database index is compared with the query index in order to discard graphs of the database not containing some features present in the query graph. This phase allows to build a set of graph candidates which will be verified in the matching phase through a sub-graph matching algorithm. The main drawback of systems based on the above architecture is the index size, construction time and the filtering time. Therefore, compact and expressive representations of such index are needed.

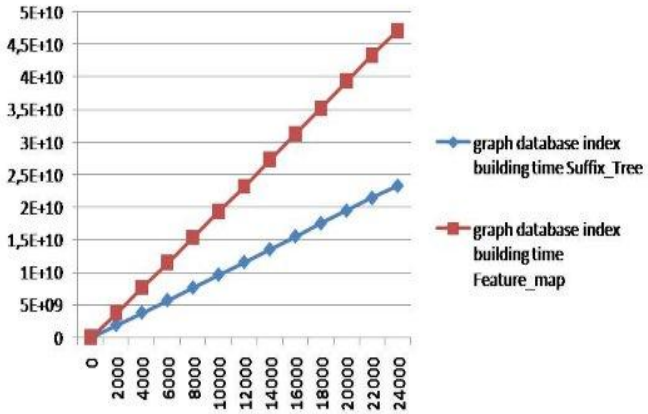
Methods

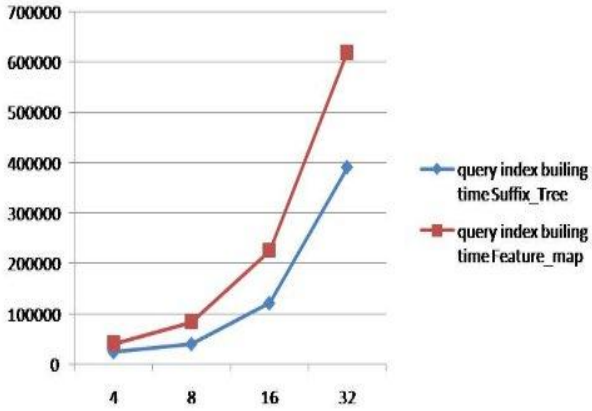
In this work a compact representation of the GraphGrep index is proposed. In GraphGrep, the feature space is composed by paths of length up to 4. Therefore, paths of greater length contain smaller paths. Moreover, graphs usually have a not large label space (e.g. chemical molecules), thus, the same partial combination of labels could be present several times in the features of different graphs. Such characteristics of the feature space allow its natural organization through a Suffix tree. Then, by exploiting the paths sharing the same prefixes the redundancy of the index is easily reduced. Although such a representation is very natural and simple it is able to speed up either the index construction and the filtering phase.

Results

The system has been tested using the Antiviral Screen Dataset. The AIDS database contains the topological structures of 42,000 chemical compounds that have been tested for evidence of anti-HIV activity. It contains sparse graphs having from 20 to 270 nodes. Queries were extracted at random from the AIDS database. Indexing paths with respect to subgraphs may result more expensive in preprocessing time and indexing space. However, it has been proved that paths have better filtering end querying time. Results show that a further improvement on path-index base system is achieved by making use of Suffix Trees. The figure depicts (graph database and query) index construction and filtering time using a subset of 24000 molecules. The query size ranges from 5 to 32 edges. The query and the filter time are the average time over 100 queries of each size. Experiments clearly show that the Suffix Tree representation of the feature space results much faster than the GraphGrep indexing (called here feature map) in terms of both index constructions and filtering time. Both indexing methods require the same amount of space.







Time is reporting as number of clocks (number of clocks per second:14318180). Algorithms are implemented in C++ under Linux OS.

Contact : giugno@dmi.unict.it