Chapter 9

TEDI: Efficient Shortest Path Query Answering on Graphs

Fang Wei
University of Freiburg, Germany

ABSTRACT

Efficient shortest path query answering in large graphs is enjoying a growing number of applications, such as ranked keyword search in databases, social networks, ontology reasoning and bioinformatics. A shortest path query on a graph finds the shortest path for the given source and target vertices in the graph.

Current techniques for efficient evaluation of such queries are based on the pre-computation of compressed Breadth First Search trees of the graph. However, they suffer from drawbacks of scalability. To address these problems, we propose TEDI, an indexing and query processing scheme for the shortest path query answering. TEDI is based on the tree decomposition methodology. The graph is first decomposed into a tree in which the node (a.k.a. bag) contains more than one vertex from the graph. The shortest paths are stored in such bags and these local paths together with the tree are the components of the index of the graph. Based on this index, a bottom-up operation can be executed to find the shortest path for any given source and target vertices.

Our experimental results show that TEDI offers orders-of-magnitude performance improvement over existing approaches on the index construction time, the index size and the query answering.

DOI: 10.4018/978-1-61350-053-8.ch009
INTRODUCTION

Querying and manipulating large scale graph-like data have attracted much attention in the database community, due to the wide application areas of graph data, such as ranked keyword search, XML databases, bioinformatics, social network, and ontologies. The shortest path query answering in a graph is among the fundamental operations on the graph data.

In a ranked keyword search scenario over structured data, people usually give scores by measuring the link distance between two connected elements. If more than one path exists, it is desirable to retrieve the shortest distance between them, because shorter distance normally means higher rank of the connected elements (Tran, T., Wang, H., Rudolph, S. & Cimiano, P. (2009), He, H., Wang, H., Yang, J. & Yu, P. S. (2007)). In a social network application such like Facebook, registered users can be considered as vertices and edges represent the friend relationship between them. Normally two users are connected through different paths with various lengths. The problem of retrieving the shortest path among the users efficiently is of great importance.

Let $G = (V, E)$ be an undirected graph where $n = |V|$ and $m = |E|$. Given $u, v \in V$, the shortest path problem finds a shortest path from $u$ to $v$ with respect to the length of paths from $u$ to $v$. One obvious solution for the shortest path problem is to execute the Breadth First Search (BFS) on the graph. The time complexity of the BFS method is $O(n)$. If the graph is of large size, the efficiency of query answering is expected to be improved.

The query answering can be performed in constant time, if the BFS tree for each vertex of the graph is pre-computed. However, the space overhead is $n^2$. This is obviously not affordable if $n$ is of large value. Therefore, appropriate indexing and query scheme for answering shortest path queries must find the best trade-off between these two extreme methods.

Xiao et al. (Xiao, Y., Wu, W., Pei, J., Wang, W. & He, Z. (2009)) proposed the concept of compact BFS-trees where the BFS-trees are compressed by exploiting the symmetry property of the graphs. It is shown that the space cost for the compact BFS trees is reduced in comparison to the normal BFS trees. Moreover, shortest path query answering is more efficient than the classic BFS-based algorithm. However, this approach suffers from scalability. Although the index size of the compact BFS-trees can be reduced by 40% or more (depending on the symmetric property of the graph), the space requirement is still prohibitively high. For instance, for a graph with 20 K vertices, the compact BFS-tree has the size of 744 MB, and the index construction takes more than 30 minutes. Therefore, this approach cannot be applied for graphs with larger size that contain hundreds of thousands of vertices.

OUR CONTRIBUTIONS

To overcome these difficulties, we propose TEDI (Tree Decomposition based Indexing) (Wei, F. (2010)), an indexing and query processing scheme for the shortest path query answering. Briefly stated, we first decompose the graph $G$ into a tree in which each node contains a set of vertices in $G$. These tree nodes are called bags. Different from other partitioning based methods, there are overlapping among the bags, i.e., for any vertex $v$ in $G$, there can be more than one bag in the tree which contains $v$. However, it is required that all these related bags constitute a connected subtree (see Definition 1 for the formal definition).

Based on the decomposed tree, we can execute the shortest path search in a bottom-up manner and the query time is decided by the height and the bag cardinality of the tree, instead of the size of the graph. If both of these two parameters are small enough, the query time can be substantially improved. Of course, in order to compute the shortest paths along the tree, we have to pre-compute...
Related Content

An Analysis of Semantic Overload in Database Access Systems Using Multi-table Query Formulation
[www.igi-global.com/article/analysis-semantic-overload-database-access/51176?camid=4v1a](www.igi-global.com/article/analysis-semantic-overload-database-access/51176?camid=4v1a)

Simplifying the Formulation of a Wide Range of Object-Oriented Complex Queries
[www.igi-global.com/article/simplifying-formulation-wide-range-object/3250?camid=4v1a](www.igi-global.com/article/simplifying-formulation-wide-range-object/3250?camid=4v1a)

Metric Databases
[www.igi-global.com/chapter/metric-databases/11174?camid=4v1a](www.igi-global.com/chapter/metric-databases/11174?camid=4v1a)

Temporal Aggregation Using a Multidimensional Index
[www.igi-global.com/article/temporal-aggregation-using-multidimensional-index/3371?camid=4v1a](www.igi-global.com/article/temporal-aggregation-using-multidimensional-index/3371?camid=4v1a)