A RDBMS based Efficient Method for Shortest Path Searching over Large Graphs considering Constraint Conditions

Jihye Hong, Yongkoo Han, Young-Koo Lee

Department of Computer Engineering, Kyung Hee University, Yongin-si, Gyeonggi-do, Korea
{hjhh, ykhan, yklee}@khu.ac.kr

Abstract. Nowadays, graphs are used in many applications and the graph size is rapidly increasing. The shortest path discovery considering constraint conditions is very important. For example, when a road is temporarily not available, paths not including the invalid road are required. In this paper, we propose an efficient RDBMS based method that searches a shortest path considering constraint conditions in a large graph. The proposed method finds the shortest path not containing the constraint path by avoiding expanding the paths including the given constraint path. For improving the performance, we propose an efficient expanding strategy that uses the index table proposed by the FEM framework. Experiments show that the proposed method can achieve high space efficiency in a large graph.

Keywords: Shortest Path Searching, RDBMS, Graph Searching, Constraint Condition.

1 Introduction

Graphs provide a way to represent relationships between objects, where some pairs of objects are connected by links. Graphs are used in many applications such as social networks, web page links, transportation networks, etc. Nowadays, graph sizes are rapidly increasing, and the graphs cannot fit in the memory. Therefore, in-memory graph operations have degradation in performance or cannot work well.

Recently, RDB based shortest path and subgraph discovery methods [1-3] have been proposed. Gao et al. [3] proposed a generic Frontier-Expansion-Merge (FEM) framework for graph search operations in the relational database (RDB) context. RDB is an effective data management approach for large scale and complex data. In order to improve the performance of shortest path discovery, the FEM framework uses an index table that stores pre-computed partial shortest paths having shorter distances than the given threshold for all nodes. However, there is no existing work considering constraint conditions in RDB.

The shortest path discovery considering constraint conditions is very important since there are many applications in real-world. For example, when a road is temporarily not available, paths not including the invalid road are required. Similarly,
this kind of constraint can occur in many networks such as social networks, web page links, and telephone networks. However, it is difficult to find a shortest path with constraint conditions on the FEM framework because the index table does not directly provide all of the path segments including the constraint path.

In this paper, we propose an efficient RDB based method that finds a shortest path with constraint conditions in a large graph. The proposed method searches the shortest path not containing the constraint path by avoiding expanding the paths including the given constraint path. We propose an efficient expanding strategy that uses the index table proposed by the FEM framework. From the experimental results, we show that the proposed method can find a shortest path with high space efficiency in a large graph.

2 Related Work

RDB based shortest path and subgraph discovery methods [1-3] have been proposed. RDB is an effective data management approach for large scale and complex data because RDB provides a stable infrastructure and several graph related functions such as breadth-first-search (BFS) and graph reachability operations. HDB-SUBDUE[1] and DB-FSG[2] proposed a RDB based frequent subgraph mining method. Gao et al. [3] proposed a generic Frontier-Expansion-Merge (FEM) framework for graph search operations using three corresponding operators in the relational database (RDB) context.

Memory based shortest path searching methods considering constraint conditions [4-5] have been proposed. Ahmed et al. [4] proposed an incremental algorithm for constructing shortest path searching tree that reflects constraint paths. Villeneuve et al. [5] proposed a method to avoid the constraint path by pre-computing k-shortest paths. These memory based approaches cannot be applied for large-scale graphs. Moreover, to the best of our knowledge, there is no existing work considering constraint conditions in RDB.

3 A RDBMS based Graph Searching Framework

Graph searching is widely used for graph algorithms finding specific subgraphs such as the shortest path between two nodes and graph reachability. Most of graph searching algorithm show a generic search process that iteratively extends nodes having results of query with high possibilities. For efficient implementing the generic search process, the FEM framework proposed three basic SQL operators: Frontier operator, Expand operator, and Merge operator. F-operator selects next expanding nodes, called frontier nodes, from the visited nodes. E-operator expands the frontier nodes. M-operator merges the new expanded nodes into the visited nodes.

Shortest path searching generally adopts breadth first search (BFS) to traverse a graph. BFS can only reduce the search space in the case that shortest path has a small number of nodes. A large-scale graph must have a long shortest path. Therefore, BFS requires a large number of iterative expansions in a large scale graph.
The FEM framework also requires BFS to expand all of edges of frontier nodes. For the efficient searching, the FEM framework pre-computes path segments having shorter distances than the given threshold for all nodes and stores the path segments into an index table called SegTable. By expanding the path segments in the SegTable, we can meet the termination condition of the searching quickly and ensure to reduce the iterations for expanding paths.

4 A RDBMS based Shortest Path Searching Method Considering a Constraint Condition

Definition 1. Constraint path Given the shortest path, \( p_s = s \rightarrow n_1 \rightarrow \cdots \rightarrow n_k \rightarrow t \). We call a path \( p \) an constraint path if \( p \) cannot be included in the shortest path. We denote the constraint path as \( p^c \).

We find the shortest path not containing the constraint path by avoiding expanding the paths including the given constraint path. For avoiding expanding these path including the constraint, a naive approach is to keep all the edge information of each path segment. However, this approach is inefficient since it requires additional \( m \times n \) times space, where \( m \) is the average number of path segments per each edge and \( n \) is the number of edges. The larger index space requires more I/O cost.

In order to avoid the paths containing the constraints, we exploit the index table approach proposed in the FEM framework rather than keeping all the edge information of each path segment. The index table consists of starting node, destination node, parent node of destination node, distance of the path segment. It reduces unnecessary iterations by expanding original edges and path segments together.

However, the index table does not directly provide all the path segments including the constraint path. We find all path segments of the constraint path in the index table. The distances of the path segments of the constraint path are updated as infinite values for avoiding expanding these path segments. Therefore, the proposed approach does not use additional index space. We can efficiently compute a constraint query without modifying original algorithm and framework.

We give the complete algorithm of searching all segments including a constraint path \( p^c = u \rightarrow \cdots \rightarrow v \). First, we find all the segments having distance shorter than \( l_{thd} \) from \( u \) where \( l_{thd} \) is the index threshold and \( p^c \) is the constraint path. Second, we store all the segments found in first step on FwdSegments table. Similar to the second step, we can store the segments from \( u \) in BckSegments table. Third, we combine one segment in FwdSegments and another segment in BckSegments. In this way, we can consider all combinations having distance of combined segment shorter than \( l_{thd} \). If a distance is satisfied with the given threshold, we insert the combined segment into the output set.
5 Experiments

In this section, we experimentally evaluate the effectiveness and the efficiency of our relational approach on a real dataset.

We use scientific collaboration networks (SCN) dataset [6] for the tests. SCN consists of about 170,000 edges. A range of edge labels is from 0.03 to 9.9 and the index threshold $l_{\text{thd}}$ is set to 6. We conduct experiments over a commercial database system. In order to show the efficiency of our method, we have implemented a naïve approach that keeps all the edge information of each path segment (SegPath) and compare the execution time and space costs. The constraint paths for the experiment are randomly selected.

We randomly generate 100 shortest path queries with constraint paths for comparing the average time cost. We use the single directional Dijkstra’s approach [3] using the index table for searching shortest path. In Figure 1(a), the proposed method (ConstSegment) requires about 40K rows and SegPath requires about 1,300K rows. We also randomly generate 100 shortest path queries with constraint path for comparing the average time cost. In Figure 1(b), ConstSegment consumes 0.75 seconds and SegPath consumes 0.72 seconds. These methods show similar time cost. From the experimental results, ConstSegment shows higher space efficiency than SegPath with similar execution time.

![Fig.1. Comparisons of space and time costs.](image)

6 Conclusion

An RDB based efficient method has been proposed for shortest path searching considering a constraint path in a large-scale graph. The proposed method finds the shortest path not containing the constraint path by avoiding expanding the paths including the given constraint path. Experiments show the proposed method requires smaller disk space than a naïve method with similar execution time.
Acknowledgements. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2013R1A2A1A05056375).

References