SHORTEST PATH ESTIMATION FOR GRAPH DATA USING A RELATIONAL TECHNIQUE

Priyanka Naphade and Yashawant Dongre

Department of Computer Engineering, VIIT, Pune, India

ABSTRACT

Recently graph data rises in many applications and there is need to manage such large amount of data by performing various graph operations over graphs using some graph search queries. Many approaches and algorithms serve this purpose but continuously require improvement over it in terms of stability and performance. Such approaches are less efficient when large and complex data is involved. Applications need to execute faster in order to improve overall performance of the system and need to perform many advanced and complex operations. Shortest path estimation is one of the key search queries in many applications. Here we present a system which will find the shortest path between nodes and contribute to performance of the system with the help of different shortest path algorithms such as bidirectional search and AStar algorithm and takes a relational approach using some new standard SQL queries to solve the problem, utilizing advantages of relational database which solves the problem efficiently.

KEYWORDS

Relational database, graph search queries, shortest path.

1. INTRODUCTION

Many Graph Search queries are in existence to perform operations over graphs. Nowadays Facebook makes use of graph search for connectivity purposes. Graph search suffers from many challenges such as how to rank the results etc. Applications are in existence for providing answers to specific queries using different search algorithms or search queries. For Instance, Reachability Query is useful for deciding whether there is path between two nodes or not. Shortest path query gives the minimum distance between two nodes. Recently due to increasing use of Web, abundant amount of data emerges out which usually be in the form of graph data. As the use of social networking sites goes on increasing vast amount of graph data emerges and goes on increasing on very fast order so it is difficult to store it in main memory. Many existing approaches are there for operations over graphs but often these approaches come across some difficulties regarding memory so need enhancement over it.

In case of social networking, many social network graphs can be prepared and operations are performed on these graphs for finding solutions to queries. For example, how two individuals are connected or how far or close these individuals are by structuring various online social networks into specific structures thus making shortest path query one of the basic and important query especially in social networking sites where interaction among individuals is more important. Social data can be can be inspected by accumulating information from four prominent online informal organizations: Flickr, YouTube, Live Journal, and Orkut. [1]. Again this query is useful

DOI : 10.5121/ijdms.2015.7403

in road networks where user wants to find minimum possible distance between any two locations. Because of abundant amount of graphs and interest in graph operations, many seemingly straightforward operations become challenging. Graphs can be stored in graph databases such as neo4j [2] that use graph data structure with nodes, edges and properties to represent data and store data. But need some substantial efforts for preserving graphs and for improving performance. In this paper, we turn our attention to the computation of shortest paths between any two nodes in the graph, a problem with long algorithmic history.

Ruiwen Chen proposed strategies of Relational database to manage massive graphs with the help of framework which combines horizontal partitioning and graph partitioning with clustering algorithms minimizing access operations[3]. Graphs are frequently used in today's automated world in a lots of applications, such as online social networks (like LinkedIn, Facebook, MySpace), entity-relationships in large-scale knowledge repositories ,for interaction models in the field of biology, transportation networks, the massive hyperlink graph between documents of the World Wide Web, XML data, VLSI design in electronics and many more. Let G = (V, E) denotes a graph consist of V nodes and E edges. An edge is represented by e= (a, b) where a, b Ɛ V.A path R from a_0 to a_x is denoted by $a_0 \rightarrow a_x$ which is denoted as ($a_0 \rightarrow a_1$), ($a_1 \rightarrow$ a2), ($a_{x-1} \rightarrow a_x$). Between a and b shortest path is represented by δ (a, b).

Relational Database can be used to manage many complex operations such as data mining, statistical operations that show the power of relational database Many algorithms are in existence for shortest path query such as Dijkstra's [4] , AStar algorithm. AStar algorithm is an extension of Dijkstra's algorithm which is considered as more efficient in terms of performance. Astar is one of the many search algorithms used in graph traversal that takes an input, evaluate a number of possible paths and returns an optimal solution. AStar algorithm is applied to FEM Framework [25] to find shortest path between nodes. AStar algorithm finds shortest path between nodes having negative weights providing optimal solution. In case of Euclidian graphs AStar algorithm takes Euclidian distance between two nodes as an input and based on these distances path is computed.

2. RELATED WORK

Relational Database can be used to handle graph data using some new features of SQL such as window function and merge statement can improve the articulation as well as enhance the execution of the proposed system. Authors have proposed two enhancement methodologies particular to shortest path disclosure inside the FEM system. In the first place bi directional Dijkstra's algorithm is used in the path finding, diminishing the search space. Second, execution is enhanced with the help of an index named SegTable [5].

Many Literatures describes external graph operations. Bahmani et al. [6] discussed a Map Reduce algorithm for Monte Carlo close estimation of personalized Page Rank vectors of all nodes in a graph achieving high scalability. Fang Wei [7] proposed a tree decomposition approach for shortest path computation and uses bottom-up approach for finding shortest path in linear time and uses tree decomposition algorithms that can be applied to large graphs which can fit into main memory.

Some approaches works on graphs in compressed form. In [8] Srihari et al. proposed a straightforward lightweight structure that uses graph applications along with the RDBMS.A SQL

based framework is proposed to mine large graphs and processing on graphs which is mainly applicable to transactional datasets due to gap between relational context and graphs. It Combines graph applications with relational database for storing large graphs and uses SQL queries for querying and accessing of data.

Aggarwal et al. [9] propose an approach to obtain minimum-cuts on sampled edges using a connectivity index. In [10] Mayfield et al. present ERACER, an iterative statistical structure that preserve data and correct errors automatically, if any based on relational dependency network, Inference algorithms. Hutchinson et al. [11] proposed an approach for storing large planer huge graphs to store in memory. When Graph size increase tremendously then it not possible for internal memory to store it. PDM, An external shortest path index is proposed on planar graphs.
In [12] proposed a novel thought, H∗-graph, which characterizes the centre of a system and augments to envelop the area of the centre for MCE calculation. Cheng et al. propose the first external-memory algorithm for MCE that uses the H∗-graph to bind the memory utilization. Far reaching investigations confirm that algorithm proficiently forms substantial systems that can't be fit in the memory.

Many approaches take into account various indexes such as Landmark index. M. Potamias et al. [13] provided solution for calculating shortest path in a large network and used Landmarks indexes. It gives more accurate results particularly in less time using Random, Degree, Centrality strategies. At runtime, when the distance between a pair of nodes is required, it can be evaluated rapidly by consolidating the pre-computed distances. Miao Qiao et al. [14] proposed a local landmark indexing approach based on graph embedding techniques. Optimization is achieved using graph embedding strategies along with triangle inequality property, querying using relational operators that particularly work on compressed graphs. Using this strategy Indexing overhead is minimized. E. Cohen et.al [15] used 2-hops index which considers a few nodes which then considered as landmarks for optimization purpose but time required is too large for practical limits.

M. Benedikt et.al present a system that takes into account a novel idea of attribute translation grammars (ATGs) that extend DTD using some semantic rules and SQL queries[16]. This idea is used for optimization purpose using better evaluation plans. S. Trißl et.al presented the GRIPP list structure by combining relational and graph operations (Graph Indexing based on Pre and Post order numbering) for answering and processing in graphs such as reachability query and executed in linear constant time and space only[17].

D. Wagner et al. [18] proposed many techniques for improving performance of computation of shortest path, such as Bidirectional search which is an extension to Dijkstra's algorithm, Goal-Directed Search that helps keep up the accuracy of the algorithm. Goal Directed search that depends upon available space and time required to perform some preliminary operations. Bidirectional search proceeds by searching path in both directions i.e. forward and backward and path is calculated when forward search meets backward search. Goal Directed techniques proceed by considering priorities of the nodes and changing that priorities by adding heuristics to cost of the node. Moreover, they talk about how combinations of speed-up methods can be acknowledged to take advantage from distinctive procedures.

Kriegel et al. [19] proposed a hierarchical reference node embedding approach that achieves better scalability by arranging reference nodes in multiple levels. An index structure is proposed

for mining patterns for graphs which are too big to fit into main memory [20]. An algorithm is proposed ADI-Mine, which is used to build the index structure that contribute to scalability of graph mining.

Relational database have been proved to support several data types over years. F.Tian et al. [21] proposed an XML publish/subscribe system using a relational approach. Candidates in a publish/subscribe-based communication system can act as a publisher or a subscriber of information. Publishers produce information sometimes called notifications, which is consumed by candidates which act as subscribers. A subscriber or consumer expresses their interests with the help of subscriptions for particular notifications. Relational database is used for matching purposes and evaluation of subscriptions is done using relational operations. Shanmugasundaram et al. [22] proposed a system that performs various operations on XML data such as transformation of semi-structured queries into relational operators and SQL queries which helps to manage data efficiently.

3. PROPOSED SYSTEM ARCHITECTURE

3.1. Problem Definition

An optimization of performance of relational database system using shortest path algorithm.

In graph theory various types of graph algorithms are present, for shortest path finding problem solved by using Best first search algorithms and Breadth first search algorithm etc. Bidirectional algorithm is used to find shortest path with the help of FEM framework using relational database. Further optimization of relational database is done using AStar algorithm and indexing on tables along with FEM Framework [25]. Basically this algorithm is used for gaming purpose.

The proposed system is used to optimize the performance of the relational database system with the help of new effective algorithms and by using indexing. Graph search is accomplished using a basic generic framework which proceeds using selection, expansion and merging operations done with the help of SQL statements. After initializing the visited node's set it repeatedly starts search. Visited nodes include all nodes that are involved in the searching process. Frontier nodes are from these visited nodes. Expanded nodes are obtained in the next iteration from the frontier nodes and marked as new visited nodes. These newly visited nodes in the next iteration are obtained by merging expanded nodes and earlier visited nodes. The System architecture for finding shortest path is shown in the figure 1.

Initially a Dataset file is selected which contains source and destination nodes and pre-processing is done on that file by randomly assigning weights to these edges. If graph is the input then store it in a dataset file and then perform pre-processing on that file. This file is stored in database after pre-processing. Users provide Inputs to system that run A* algorithm along with FEM framework. Algorithm starts by taking input as nodes and edges of graph G= (V, E) that are stored in form of Input table and output table. In the first step source node is initialized in the source table using SQL query. Then until table is not empty next frontier node's ID is found and paths are expanded using this node ID by grouping similar nodes and arranging them in the ascending order of their weights.

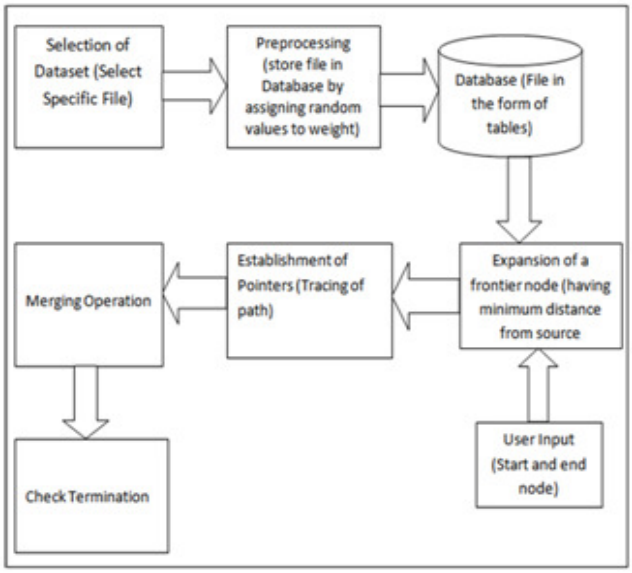

Figure 1. System Architecture

If there are no affected tuples or when the dead end is meet, then terminate and finalize that node. If the target node is found then terminate and after detecting the edges in the shortest path return the shortest path. Output will be the shortest path between the nodes.

3.2 Algorithm:

1. Create a Source Table and put only source node S to the Source table, OPEN.
2. Create an Output Table, CLOSED initially having no node.
3. While OPEN is not empty, Go to step 4.
4. Select top node from OPEN and keep it on CLOSED table, which is said to be a frontier node n_f having minimum distance from source.
5. If n_f is a target node or dead end meet, Return a path from n_f to S.
6. Expand node n_f and Create table M that contains unvisited nodes of n_f which are not already parents of n_f. Arrange Records in Ascending Order of weights of the nodes and store it in temporary table.
7. Merge nodes obtained after step 5 into the CLOSED table. If these new nodes have minimum distance from start node then add these nodes to CLOSED table replacing earlier nodes. Otherwise directly add to CLOSED.

Trace the path by finalizing parents of nodes to obtain the shortest path.

4. RESULTS AND DISCUSSION

Traditional AStar algorithm works by exploring all possible paths from a node to a goal node and when obstacle is encountered it changes its path. When FEM framework [25] is used along with it one node will be selected as a frontier node and all nodes related to that node are explored reducing number of possibilities and search continues in one direction. In bidirectional search, one table is partitioned into different tables so significant amount of time will be consumed in

partitioning the table. While finding paths we have to take aggregated results by using collective values from all tables with the help of various joins making it more expensive. But need to visit less number of nodes saving memory.

We have implemented proposed system on personal computer having configuration i3 processor 2.20 GHz, 4GB RAM etc. We have used Net Beans IDE 8.1.0 and SQL Server 2008 tools for implementation of improved AStar algorithm in java.

We conduct few experiments for testing of proposed system results with the results in [25]. The experiments contain dataset with different sizes i.e. small dataset and large dataset. The smaller dataset contains 100 to 1000 records; larger dataset contains up to 500000 records. The figure 2. Show the execution time for computation of shortest path over Road Net dataset containing more than 500000 records in RDBMS. This graph shows that our improved AStar algorithm is efficient than the bidirectional algorithm due to indexing on table. The main problem with bidirectional algorithm is it consumes more time in partitioning the graph. However, our system is able to overcome the problems with bidirectional algorithm. The obtained results in figure 3 shows that our algorithm works even better for smaller datasets i.e. dataset containing up to 1000 records.

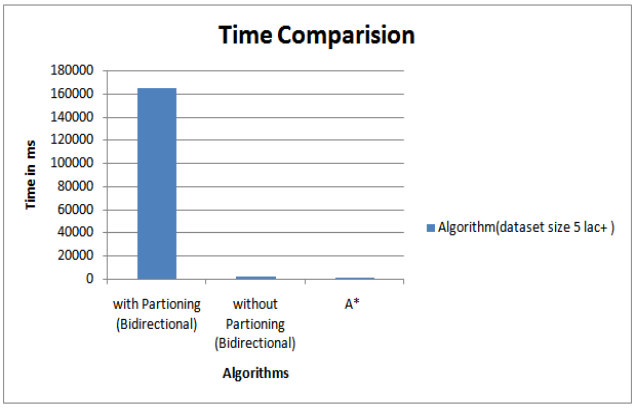

Figure 2: Time comparison between Bidirectional search and Proposed technique for dataset size 5lac+

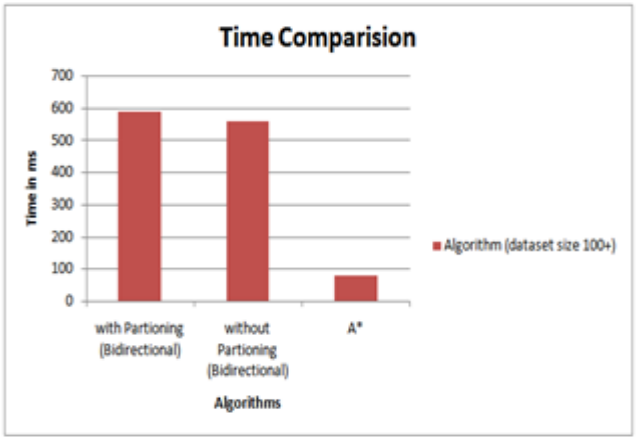

Figure 3: Time comparison between Bidirectional search and Proposed technique for dataset size 100+

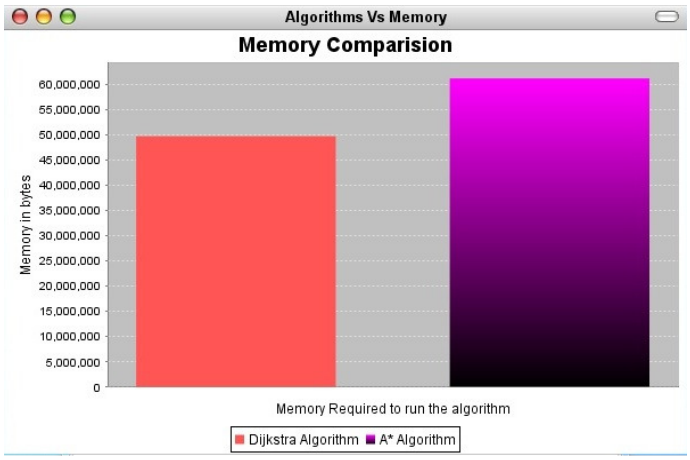

Figure 4: Memory comparison between Bidirectional search and Proposed Technique

In terms of memory efficiency, our system requires more memory than the system proposed in [25]. The memory require for computation of shortest path , but it is efficient than the other shortest path algorithms mentioned in related work section.

5. CONCLUSION

In this paper we introduced an approach for finding shortest path using relational method and AStar algorithm. We have done experiments on two different datasets with Bidirectional approach with and without partition. From the results of experiments on two varying datasets, we came to a conclusion that our new approach i.e. AStar algorithm along with FEM framework is more efficient than existing approach and requires more memory than existing approach. An algorithm for finding shortest path in relational context is developed.

We proposed an efficient algorithm that uses optimization strategy with indexing that increase performance of the system. Proposed system requires less execution time than Bidirectional Search. Proposed system gives better results than bidirectional search, without Partitioning.

ACKNOWLEDGEMENTS

The authors wish to thank everyone who has contributed to this work directly or indirectly.

REFERENCES

[1] A. Mislove, M. Marcon, K.P. Gummadi, P. Druschel, and B. Bhattacharjee, “Measurement and Analysis of Online Social Networks,” Proc. Seventh ACM SIGCOMM Conf. Internet Measurement (IMC ’07), Oct. 2007.
[2] “Neo4j,” http://neo4j.org/, 2014.
[3] Ruiwen Chen, “Managing Massive Graphs in Relational DBMS"IEEE International Conference on Big Data.,2013.
[4] E. Dijkstra, “A Note on Two Problems in Connexion with Graphs,” Numerische Mathematik, vol. 1, pp. 269-271, 1959.

[5] J. Gao, R. Jin,J. Zhou, J. Xu, X. Jiang, and T. Wang,"Relational Approach for Shortest Path Discovery over Large Graphs," Proc. VLDB Endowment, vol. 5, no. 4, pp. 358-369, 2011.
[6] B. Bahmani, K. Chakrabarti, and D. Xin, "Fast Personalized Pagerank on Mapreduce," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '11), pp. 973-984, 2011.
[7] Fang Wei, "TEDI: Efficient Shortest Path Query Answering on Graphs", ACM 978-1-4503- 0032, 2010.
[8] S. Srihari, S. Chandrashekar, and S. Parthasarathy, "A Framework for SQL-Based Mining of Large Graphs on Relational Databases," Proc. 14th Pacific-Asia Conf. Advances in Knowledge Discovery and Data Mining—Volume Part II (PAKDD'10), pp. 160-167, 2010.
[9] C. Aggarwal, Y. Xie, and P. Yu, "GConnect: A Connectivity Index for Massive Disk-Resident Graphs," Proc. VLDB Endowment, vol. 2, no. 1, pp. 862 873, 2009.
[10] C. Mayfield, J. Neville, and S. Prabhakar, "ERACER: a Database Approach for Statistical Inference and Data Cleaning," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIMGMOD '10), pp. 75-86, 2010.
[11] D. Hutchinson, A. Maheshwari, and N. Zeh, "An External Memory Data Structure for Shortest Path Queries," Discrete Applied Math., vol. 126, pp. 55-82, no. 1, 2003.
[12] J. Cheng, Y. Ke, A.W. Fu, J.X. Yu, and L. Zhu, "Finding Maximal Cliques in Massive Networks by H*-graph," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '10), pp. 447-458, 2010.
[13] M. Potamias, F. Bonchi, C. Castillo, and A. Gionis, "Fast Shortest Path Distance Estimation in Large Networks," Proc. Int'l Conf. Information and Knowledge Management (CIKM'09), pp. 453-470, 2009.
[14] Miao Qiao, Hong Cheng, Lijun Chang, and Jeffrey Xu Yu, "Approximate Shortest Distance Computing: A Query-Dependent Local Landmark Scheme", IEEE Transactions On Knowledge And Data Engineering, Vol. 26, No. 1, January 2014.
[15] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick, "Reachability and Distance Queries via 2-Hop Labels," Proc. 13th Ann. ACMSIAM Symp. Discrete Algorithms (SODA '02), pp. 937-946, 2002.
[16] M. Benedikt, C. Chan, W. Fan, R. Rastogi, S. Zheng, and A. Zhou, "Dtd-Directed Publishing with Attribute Translation Grammars," Proc. 28th Int'l Conf. Very Large Data Bases (VLDB '02), 2002.
[17] S. Trißl and U. Leser, "Fast and Practical Indexing and Querying of Very Large Graphs," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD'07), pp. 845-856, 2007.
[18] D. Wagner and T. Willhalm, "speed-Up Techniques for Shortest- Path Computations "Proc. 16th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA -05), pp. 156-165,), pp. 845-856, 2005.
[19] H.P. Kriegel, P. Kroger, M. Renz, and T. Schmidt, "Hierarchical Graph Embedding for Efficient Query Processing in Very Large Traffic Networks," Proc. 20th Int'l Conf. Scientific and Statistical Database Management (SSDBM '08), pp. 150-167, 2008.
[20] B. Zou, X. Ma, B. Kemme, G. Newton, and D. Precup, "Data Mining Using Relational Database Management Systems," Proc. 10th Pacific-Asia Conf. Advances in Knowledge Discovery and Data Mining ('06), pp. 657-667, 2006.
[21] F. Tian, B. Reinwald, H. Pirahesh, T. Mayr, and J. Myllymaki, "Implementing a Scalable XML Publish/Subscribe System Using a Relational Database System," Proc. ACM SIGMOD Int'l Conf.Management of Data (SIGMOD '04), pp. 479-490, 2004.
[22] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. DeWitt, and J. Naughton, "Relational Databases for Querying XML Documents: Limitations and Opportunities," Proc. 25th Int'l Conf. Very Large Data Bases (VLDB '99), pp. 302-314, 1999.
[23] R. Ronen and O. Shmueli, "SoQL: A Language for Querying and Creating Data in Social Networks," Proc. IEEE 25th Int'l Conf. Data Eng. (ICDE'09), pp. 1595-1602, 2009.
[24] A. Goldberg and C. Harrelson, "Computing the Shortest Path: Search Meets Graph Theory," Proc. 16th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '05), pp. 156-165, 2005.
[25] Jun Gao, Jiashuai Zhou, Jeffrey Xu Yu, and Tengjiao Wang,"Shortest Path Computing in Relational DBMSs," IEEE transactions on knowledge and data engineering, vol. 26, no. 4, APRIL 2014.

AUTHORS

Ms. Priyanka S. Naphade is a post graduate student in Computer Engineering from Savitribai Phule Pune University, Pune, Maharashtra State, India

Prof. Yashawant V. Dongre is Assistant Professor in Vishwakarma Institute of Information Technology (VIIT) at Savitribai Phule Pune University, Pune, and Maharashtra State, India. His area of interest includes database management, data mining and information retrieval. He has several journal papers to his credit published in prestigious journals.