A Brief Survey on Distributed Graph Algorithms for Shortest Distance

V. Jenifer

***Abstract*---**There is an extended history of study in theoretical computer science faithful to designing proficient algorithms for graph problems. In several modern applications the graph in query is altering over time, and to avoid rerunning algorithm on the entire graph every time a small change occurs. This paper aims to present a brief survey on graph theory based on Shortest Distances in Dynamic Graphs techniques in which the goal is to minimize the amount of work needed to re-optimize the solution when the graph changes. Number of relative studies namely Graph pattern matching, Spatially Induced Linkage Cognizance (SILC), Snowball Algorithm, GREEDY-SNDOP, APSP and Efficient incremental algorithms are discussed and evaluate the running time performance on the several datasets. Comparing to these algorithms the efficient incremental algorithm techniques methods outperforms having better performance than other methods.

***Keywords*---**Datamining, Dynamic Graph, Shortest Distance, Incremental Algorithms.

I. INTRODUCTION

DATA Mining is the process of finding previously an unknown patterns and trends in databases and using that information to build predictive models. Data mining combines statistical analysis, machine learning and database technology to extract hidden patterns and relationships from large databases.

In the previous years, with the advances in technology, navigation systems which helped people to get from point *X* to point *Y* as fast as possible, has also changed greatly. A few years ago, most of the navigation devices were using only pre-installed maps to determine the route. These devices converted the map paths to graphs such that nodes are the destinations in the map and the edges are the paths between them. They were updating their graphs only if a road was added to or removed from the area they are handling. The shortest path route calculated using these old devices mostly did not change for a fixed pair of source and destination points and thus they were not very efficient since they were not taking into consideration the number of cars that were using those routes and causing lots of traffic jams.

The graph framework model is similar to the normal data structure representation. The algorithm is given an original graph, and must process an online sequence of updates and queries, where the update changes the graph in some way, while the query asks for information about the current version of the graph.

There are many variations on the graph model depending on exactly what types of updates are allowed. The standard model is the fully dynamic one, where an update can insert an edge, deletes an edge, and in the case of a weighted graph it may also change the weight of an edge. The query then depends on the specific graph problem being considered. For example, in dynamic connectivity, the query LINKED(*X*, *Y*) asks whether there is a path between vertices x and y in the current version of the graph; in dynamic single source shortest paths, the query DISTANCE(*V*) asks for the shortest distance from the fixed source s to vertex v. There are some problems for which instead of queries, it is more natural to require that the algorithm maintain some substructure in the graph, such as a maximum matching or a minimum spanning tree. A common restriction of the fully dynamic setting is the partially dynamic setting where updates consist of only insertions, or only deletions. The former case is known as incremental, the latter as decremental.

There are so many existing studies are focused on shortest paths in dynamic graph particular. As before, the most general model is the fully dynamic one, where an update is allowed to insert or delete edges into the graph. In dynamic all pairs shortest paths, the query DISTANCE(*X*, *Y*) can ask for the shortest distance between any pair of vertices *X* and *Y*, while in dynamic single source shortest paths, there is a fixed source *S*, and query DISTANCE(*X*) asks for the shortest distance from *S* to a vertex *X*. All existing algorithms can be extended to find the actual shortest path, but outputting the path might by necessity take O(n) time if the path is long, so since typically want to keep the query time small, most researchers focus on answering dynamic distance queries.

This survey paper explores the Shortest Distances in Dynamic Graphs based algorithms to process sequences of edge deletions or insertions or updates and vertex deletions/insertions.

II. RELATED WORK

***W. Fan, et.al.*, (2010)** [2] proposed a class of graph patterns, in which an edge denotes the connectivity in a data graph within a predefined number of hops. In addition, to defined matching based on a notion of bounded simulation, an extension of graph simulation. To show that with this revision, graph pattern matching can be performed in cubic-time, by providing such an algorithm. It also developed algorithms for in-crementally finding matches when data graphs are updated, with performance guarantees for dag patterns.

Manuscript received on January 05, 2021, review completed on January 06, 2021 and revised on January 19, 2021.

V. Jenifer is with the KG College of Arts and Science, Coimbatore, India.

Digital Object Identifier: SE012021001.

***L. Wu, et.al.,* (2012)** [3] addressed the length of shortest path; a state-of-the-art technique was examined based on a faulty implementation that led to incorrect query results. They presented a comprehensive comparison of the most advanced spatial-coherence-based and vertex-importance-based approaches. Using a variety of real road networks with up to twenty million vertices, to evaluated each technique in terms of its preprocessing time, space consumption, and query efficiency (for both shortest path and distance queries).

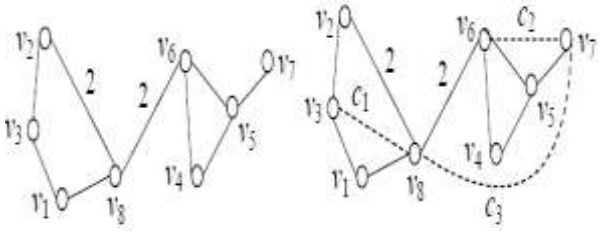

Fig.1: Road Network and Contraction Hierarchies

In figure 1 represents, a graph contains eight vertices v_1, v_2…, v_8 and nine edges. In particular, the lengths of the edges (v_2, v_8) and (v_6, v_8) equal 2, while the lengths of the other edges are 1.

***L. Planken, M. de Weerdt, and R. van der Krogt* (2012)** [4] presented present two new and efficient algorithms for computing all-pairs shortest paths. The algorithms operate on directed graphs with real (possibly negative) weights. They make use of directed path consistency along a vertex ordering d. Both algorithms run in $O(n^2w_d)$ time, where wd is the graph width induced by this vertex ordering. For graphs of constant treewidth, this yields O (n^2) time, which is optimal. On chordal graphs, the algorithms run in O(nm) time. In addition, to presented a variant that exploits graph separators to arrive at a run time of $O(nw_d^2 + n^2s)_d$ on general graphs, where $s \leq_d w_d$ is the size of the largest minimal separator induced by the vertex ordering d.

***P. Shakarian, M. Broecheler, V. S. Subrahmanian, and C. Molinaro* (2013)** [5] showed that the well-known Generalized Annotated Program (GAP) paradigm can be used to express many existing diffusion models. To define a class of problems called Social Network Diffusion Optimization Problems (SNDOPs). SNDOPs have four parts: (i) a diffusion model expressed as a GAP, (ii) an objective function want to optimize with respect to a given diffusion model, (iii) an integer k > 0 describing resources (e.g. medication) that can be placed at nodes, (iv) a logical condition that governs which nodes can have a resource (e.g. only children above the age of 5 can be treated with a given medication).

***S. S. Khopkar, R. Nagi, A. G. Nikolaev, and V. Bhembre* (2014)** [6] discussed the biggest challenges in today's social network analysis (SA) is handling dynamic data. Real-world social networks evolve with time, forcing their corresponding graph representations to dynamically update by addition or deletion of edges/nodes. The authors presented fast incremental updating algorithms along with the time complexity results for APSP, closeness centrality and betweenness centrality, considering two distinct cases: edge addition and node addition. The following time complexity results are presented:

(1) The incremental APSP algorithm runs in $O(n^2)$ time ($\square n^2$) is the theoretical lower bound of the APSP problem), (2) The incremental closeness algorithm that runs in Oðn2Þ time, and (3) The incremental betweenness algorithm runs in $O(nm + n^2 \log n)$ time.

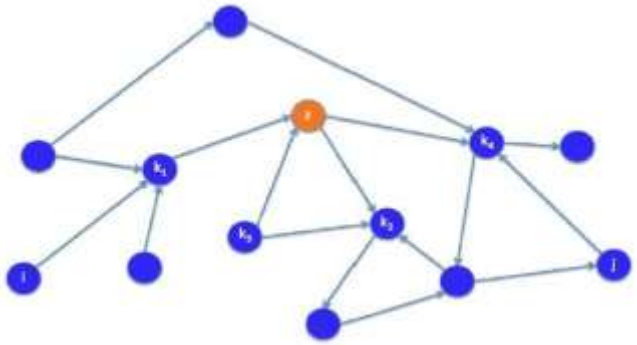

Fig. 2: Newly Added Node z shown with Neighbors k_1, k_2, k_3 and k_4.

In figure 2, shows, if any shortest path passes through the newly added node in the graph, then it must first pass through at least one of its neighbors, which were already present in the original graph.

***F. Bonchi, A. Gionis, F. Gullo, and A. Ukkonen* (2014)** [7] studied the problem of efficient approximation of shortest-path queries with edge-label constraints, for which to devise two indexes based on the idea of landmarks: distances from all vertices of the graph to a selected subset of landmark vertices are pre-computed and then used at query time to efficiently approximate distance queries.

***A. Tagarelli and R. Interdonato* (2015)** [8] discussed their objective is to push forward research in lurker mining in a twofold manner: (1) to provide an in-depth analysis of temporal aspects that aims to unveil the behavior of lurkers and their relations with other users, and (2) to enhance existing methods for ranking lurkers by integrating different time-aware properties concerning information production and information consumption actions. Network analysis and ranking evaluation performed on Flickr, FriendFeed and Instagram networks allowed us to draw interesting remarks on both the understanding of lurking dynamics and on transient and cumulative scenarios of time-aware ranking.

***S. Greco, C. Molinaro, and C. Pulice* (2016)** [9] illustrated a computing shortest distances is a central task in many graph applications. To addressed the problem of maintaining all-pairs shortest distances in dynamic graphs and propose novel efficient incremental algorithms, working both in main memory and on disk. To proved their correctness and provide complexity analyses. Experimental results on real-world datasets show that current main-memory algorithms become soon impractical, disk-based ones are needed for larger graphs, and their approach significantly outperforms state-of-the-art algorithms.

Sergio Greco, Cristian Molinaro and Chiara Pulice **(2018)** [10] discussed the number of applications dealing with dynamic graphs calls for incremental algorithms, as it is impractical to re-compute shortest distances from scratch every time updates occur. The authors addressed the problem of maintaining all-pairs shortest distances in dynamic graphs. To proposed an efficient incremental algorithms to process sequences of edge deletions/insertions/updates and vertex deletions/insertions. Their approach relies on some general operators that can be easily "instantiated" both in main memory and on top of different underlying DBMSs.

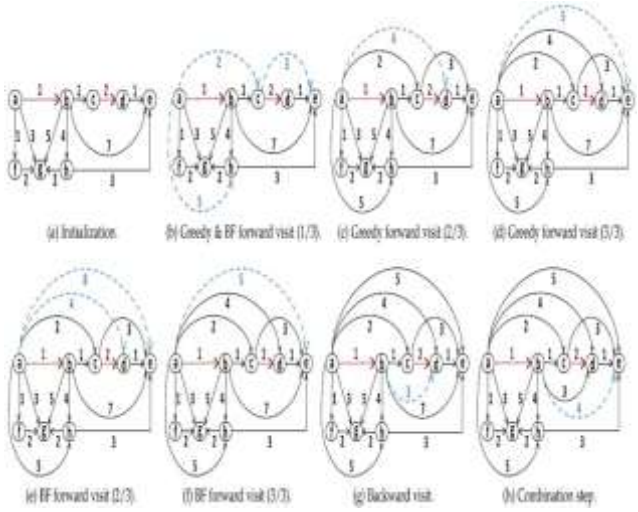

Fig. 3: Execution of Edge-Insertions-Maintenance for the Graph and the Shortest Distances

In Figure 3 illustrates how the shortest distances are updated by the two variants of Edge-Insertions-Maintenance—the inserted edges are highlighted by (red) dotted edges. The altered graph and its shortest distances are shown in figure 4.

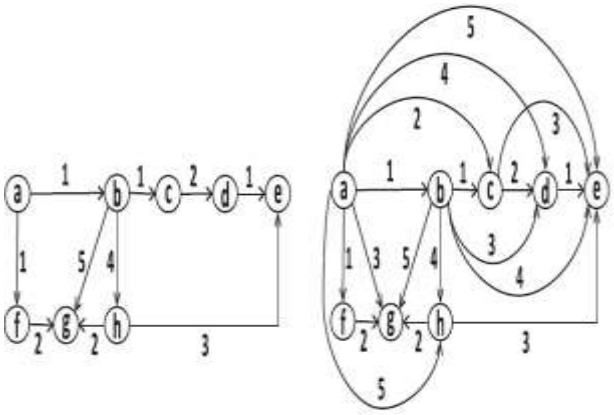

Fig. 4. Altered Graph (left) and its Shortest Distances (Right).

III. COMPARISON ANALYSIS

This survey paper aims to collect and consider papers that deal with Shortest Distances in Dynamic Graphs techniques. The objective is not to undertake a conditions review, but quite to provide a broad state-of-the-art view on these related fields. Several exisiting approaches have been projected to assist graphs, shortest distances and Maintenance, which has mentioned in a body of literature that is spread over a wide variety of applications.

TABLE 1
SUMMARY TABLE FOR COMPARISON OF SHORTEST DISTANCES IN DYNAMIC GRAPHS TECHNIQUES

Title	Algorithm	Key-Idea	Techniques	Results	Performance
Graph pattern matching: From intractable to polynomial time (2010) [2]	Graph pattern matching.	Intractable to Polynomial Time.	Incremental algorithms for DAG (directed acyclic graphs) patterns.	The algorithms are efficient and scale well with the size of data graphs, and with the size of pattern graphs.	Inc Match for Insertions performed 14 seconds.
Shortest path and distance queries on road networks: An experimental evaluation (2012) [3]	Spatially Induced Linkage Cognizance (SILC).	SILC incurs significant preprocessing time and space consumption, but it offers superior efficiency for shortest path queries.	SILC, Path-Coherent Pairs Decomposition, Contraction Hierarchies (CH), and Transit Node Routing (TNR)	The improvement comes at the cost of considerable pre-computation and space overhead.	It requires only 30 minutes to process the US dataset.
Computing all-pairs shortest paths by leveraging low treewidth (2012) [4]	Snowball Algorithm, Directed Path Consistency algorithm.	Computing all-pairs shortest paths.	Graph-theoretic techniques.	Snowball consistently outperforms than DPC.	It requires only 1000 ms to process the dataset.
Using generalized annotated programs to solve social network diffusion optimization problems (2013) [5]	Heuristic algorithm and GREEDY-SNDOP (Social Network Diffusion Optimization Problems) algorithm.	There is no polynomial-time algorithm that computes an approximation.	Logic Programming.	To solving SNDOP queries under the assumption of a monotonic aggregate function.	Runtimes of GREEDY-SNDOP is 10000 seconds.

Efficient algorithms for incremental all pairs shortest paths, closeness and betweenness in social network analysis (2014) [6]	All pairs shortest paths (APSP) algorithm.	To dynamically update by addition or deletion of edges/nodes.	Shortest path technique.	The pair is improved as a result of emergence of a new path or paths.	It takes 7000 milli seconds for random graph.
Time-aware analysis and ranking of lurkers in social networks (2015) [8]	LurkerRank algorithm.	To provide an in-depth analysis of temporal aspects.	Time-aware lurker ranking	To overcome the time-related limitation of previous formulations of lurker ranking methods.	The correlation results of 0.367 on interaction network.
Efficient maintenance of all- pairs shortest distances (2016) [9]	Incremental algorithms.	To address the problem of maintaining all-pairs shortest distances in dynamic graphs.	Single-source shortest paths and all-pairs shortest paths.	The running times suggests that maintenance after edge deletion is more expensive than maintenance after edge insertion.	It takes 100 seconds for processing entire dataset .
Efficient Maintenance of Shortest Distances in Dynamic Graphs (2018) [10]	Efficient incremental algorithms.	Sequences of edge deletions/insertions/updates and vertex deletions/insertions.	Shortest Distances in Dynamic Graphs.	The batches of 64–96 edges are due to the presence of edges with quite high influence.	The execution times for the insertion and deletion of 150 randomly chosen edges

IV. Conclusion

This paper presents a brief survey about Shortest Distances in Dynamic Graphs discussed with the different categories. This survey can be classified into dynamic graph as edge insertions, deletions or updates and vertex deletions or insertions. To concluded the discussion on dynamic graph shortest distance algorithms with edge and vertex updation by study with single and all pair paths category. It also discussed the concept of finding distance and running time execution, considers all possible datasets to a program measures which proves to be the most important criteria for Path Redundancy.

The further work enhanced and expanded for the Shortest Distances in Dynamic Graphs execution technique in parallel architecture that leverages a parallel execution algorithm.

References

[1] Xingquan Zhu, Ian Davidson, "Knowledge Discovery and Data Mining: Challenges and Realities", ISBN 978-1-59904-252, Hershey, New York, 2007.

[2] W. Fan, J. Li, S. Ma, N. Tang, Y. Wu, and Y. Wu, "Graph pattern matching: From intractable to polynomial time," Proc. VLDB Endowment, vol. 3, no. 1, pp. 264–275, 2010.

[3] L. Wu, X. Xiao, D. Deng, G. Cong, A. D. Zhu, and S. Zhou, "Shortest path and distance queries on road networks: An experimental evaluation," Proc. VLDB Endowment, vol. 5, no. 5, pp. 406–417, 2012.

[4] L. Planken, M. de Weerdt, and R. van der Krogt, "Computing allpairs shortest paths by leveraging low treewidth," J. Artif. Intell. Res., vol. 43, pp. 353–388, 2012.

[5] P. Shakarian, M. Broecheler, V. S. Subrahmanian, and C. Molinaro, "Using generalized annotated programs to solve social network diffusion optimization problems," ACM Trans. Comput. Logic, vol. 14, no. 2, pp. 10:1–10:40, 2013.

[6] S. S. Khopkar, R. Nagi, A. G. Nikolaev, and V. Bhembre, "Efficient algorithms for incremental all pairs shortest paths, closeness and betweenness in social network analysis," Social Netw. Analys. Mining, vol. 4, no. 1, 2014, Art. no. 220.

[7] F. Bonchi, A. Gionis, F. Gullo, and A. Ukkonen, "Distance oracles in edge-labeled graphs," in Proc. Int. Conf. Extending Database Technol., 2014, pp. 547–558.

[8] A. Tagarelli and R. Interdonato, "Time-aware analysis and ranking of lurkers in social networks," Social Netw. Analys. Mining, vol. 5, no. 1, pp. 46:1–46:23, 2015.

[9] S. Greco, C. Molinaro, and C. Pulice, "Efficient maintenance of allpairs shortest distances," in Proc. Sci. Statistical Database Manag., 2016, pp. 9:1–9:12.

[10] Sergio Greco, Cristian Molinaro , and Chiara Pulice, "Efficient Maintenance of Shortest Distances in Dynamic Graphs", IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 30, NO. 3, MARCH 2018.

V. Jenifer received M.Sc, M.phil degree Computer Science from RVS Arts and Science College. M.Sc (cs) which is affiliated to Bharathiar University, Coimbatore. Currently working in KGisl arts and science college. She has 2 years of collegiate teaching and 1.5 years of research experience. Her research interest includes the Data mining.