© 2016 IJSRSET | Volume 2 | Issue 5 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

Survey on Mining Shortest Path in Directed and Dynamic Graphs

B. Kaleeswari[*1], G. Boopathi [2], K. Anitha[3], [4] K. Sangeetha

[*1]Assistant professor,Computer Science and Engineering, SNS College of Technology, Coimbatore, Tamilnadu, India
[2,3,4]UG Scholar**,** Computer Science and Engineering, SNS College of Technology, Coimbatore, Tamilnadu, India

ABSTRACT

Calculating the shortest path in larger graph is one of the major problems in graph theory. There are some algorithms available for finding the shortest paths in graph. But those algorithms are not efficient for finding the shortest path in larger directed graph. So we are going to propose a new algorithm for finding the shortest path in directed and dynamic graph. The data reduction technique proxies are used for mining the shortest path in graph. The Dijktra's algorithm is used for finding the shortest path in early graph theory. But it is not efficient for larger graphs. So we are going to create a new algorithm by combining the methods of Dijktra's and data reduction techniques. By using this algorithm we are going to find the shortest path in directed and dynamic graphs.

Keywords : Graph, Shortest Path, Data Reduction Technique, Proxies

I. INTRODUCTION

Finding shortest path in graph is the fundamental problem in graph. It has the wide range of applications. For example, finding shortest path in social networks, road networks, map- reduce etc [1],[2], [3]. In social network we have to communicate between two nodes but it has a larger number of nodes connected together. In that larger network many users are sharing the resources. They are interconnected to each other. In such a large network finding a shortest path between the sender and receiver is the challenging task. Shortest path is the distance between two nodes. The distance is calculating by using several algorithms. Since 1950's computing shortest-path is have been studied. But the efficiency of those algorithm is not satisfied the level. We all seen that computing the shortest paths in larger graph consist two nodes A and B, where A is the source and B is the destination and these are nodes of a graph G. Dijktra's [4] is the one of the techniques which is used to find the shortest path in the larger graphs is a very difficult one. It is the enhanced thing of the Fibonacci heaps. But it is having the flaw that it is not applicable for the larger graphs. Because the time and space complexity for larger graphs such as social and road networks the Dijktra's algorithm is not applicable.

For example, to analyze a large graph, need often to compute closeness centrality, diameter, etc., Many of these applications require efficient computation of shortest distances. For instance, in online social networks (OSN) users may request a list of users sharing common interests. The efficiency of recommendation algorithms based on shortest distances also relies on fast distances, query.

Additional applications of this problem type exist in the logistics industry as well. For example, for long-haul trucking, the route may require multiple days of travelling long distances, necessitating the recurrent need for refuelling, lodging, eating, etc. throughout the trip. Selecting the optimal locations amongst all of these options to minimize travel time and cost is an important part of the route planning process. This basic problem is commonly used for Generalized Travelling Salesman Path Problem (GTSPP) and it is known as NP-hard, as solution. It determines not only which location type to visit, but also the optimal order in which to visit them.

For that purpose a new algorithm is developed. We are going to develop a new algorithm by combining the methods of Dijktra's algorithm and data reduction technique in directed and dynamic graph. This method is implemented by using the routing proxies and by

using the k-path shortest query algorithm. We are going to find the shortest path in road networks were the directed graph represents the one way road network. Here for road details we are going to use two dataset node set and edge set. The graph can be formed by using the details of the open World Wide map database.

II. METHODS AND MATERIAL

1. Map Database

A. Map Database Structure

There are four types of objects to describe a map in map databases as follows.

- **Node**

Each node represents a particular point in a map.

- **Node List**

A node list is an ordered list of nodes. It represents a poly line or polygon.

- **Element List**

An element list is an ordered list of nodes and node lists. This type of objects is employed to represent relationships between nodes and node lists.

- **Edge**

An edge is a relationship between two nodes. Actual map database may include the both of undirected roads and one-way roads. Then, an edge may be either *undirected* or *one-way* to represent whether a part of a road is undirected or one-way.

A map may be described using four sets of all the above four types of objects. In this paper, the two sets, a node set and an edge set, are used to evaluate *k*-shortest path query.

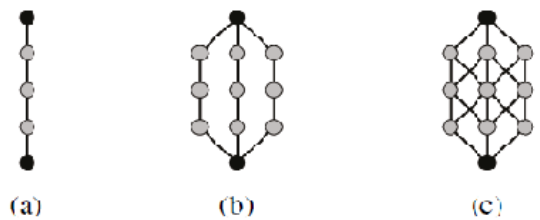


Figure 1. Example for Connected Graph

2. Graph Notions

Graph: A weighted undirected graph is defined as G(V,E,w) where V is a finite set of nodes.

Sub-graph: Graph $H(V_s,E_s,w_s)$ is a subgraph of graph G(V,E,w) That is,subgraph H simply contains a subset of nodes and a subset of edges of graph G.

Neighbours:the neighbour node v and node u if there exists an edge(u,v) and (v,u) in the graph G.
Path and Cycles:A simple path ρ has a sequence of nodes $v_1/...../v_n$,with no repeated nodes.

A simple cycle ρ is a sequence of nodes $v_1/...../v_n$ with $v_1= v_n$ and no other repeated nodes of an edge in G.

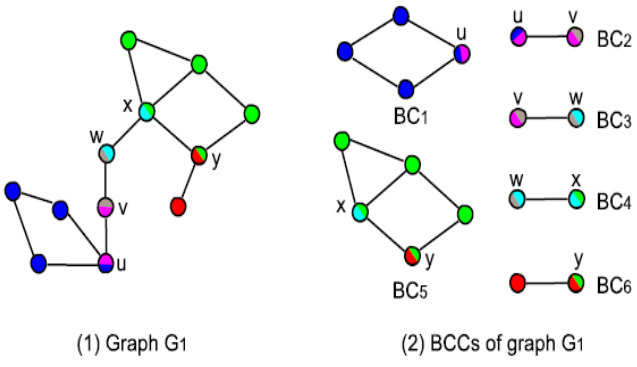
Shortest path and distances:A shortest path from one node *u* to another node *v*,denoted as (*u,v)*,is a path.The minimum length among all the paths from *u* to *v*.

The *shortest distance* between nodes *u* and *v*, denoted as *dist (u,v),* is the shortest length of all paths from *u* to *v*, the length of a shortest path from *u* to *v*.

Connected Components: A *Connected Components* of a graph is a subgraph in which any two nodes are connected by a path, and is connected to an additional nodes. A graph is connected if it has exactly one connected component, consisting of the entire graph.

Cut-nodes and Bi-connected components: A *cut-node* of a graph is a node whose removal increases the number of connected components in the graph.

A *bi-connected component* of a graph is a sub graph consisting of a maximal set of edges such that any two edges must lie on a common simple cycle.



3. Routing Proxies

In this section, we first propose routing proxies and deterministic routing areas to capture the idea of representatives and the set of nodes being represented, respectively. We then give an analysis of the properties of DRAs and their routing proxies.

We first present the notions of routing proxies and their deterministic routing areas. Proxies. Given a node u in graph GðV;EÞ, we say that u is a routing proxy (or simply proxy) of a set of nodes, denoted by Au, if and only if:

(1) node u 2 Au is reachable to any node of Au in G,
(2) allneighbors of any node v 2 Au n fug are in Au, and
(3) the size jAuj of Au is equal to or less than c,
where c is a small constant number, such as 2 or 3.
Here condition
(1) guarantees the connectivity of subgraph G½Au_, condition
(2) implies that not all neighbors of proxy u are necessarily in Au; and condition
(3), referred to as size restriction, limits the size of Au of proxy u. Intuitively, one simply checks the graph by removing u from G and its newly created CCs, and a proxy of u is a union of such CCs whose total number of nodes is at most c 1.

4. Properties of Proxies And Dras

- ✓ A small number of representatives can represent a large number of nodes in a graph
- ✓ Shortest paths and distances involved within the set of nodes being represented by the same representative can be answered efficiently
- ✓ The representatives and the set of nodes being represented can be computed efficiently.

5. Related Work

✓ Exact Methods

For exact distance queries on complex networks such as social networks and web graphs, several methods are proposed recently. Large portion of these methods can be considered as based on the idea of 2-hop cover [5]. Finding small 2-hop covers efficiently is a challenging and long-standing problem [5,6,7]. One of the latest methods is *hierarchical hub labeling*which is based on a method for road networks [1]. Another latest method related to 2-hop cover is *highway-centric labeling*[8]. In this method, we first compute a spanning tree T and use it as a "highway". That is, when computing distance dG(u, v) between two vertices u and v, we output the minimum over dG(u,w1)+dT (w1,w2)+dG(w2, v) where w1 and w2 are vertices in labels of u and v, respectively, and dT (・, ・) is the distance metric on the spanning tree T . An approach based on *tree decompositions* is also reported to be efficient [9,10]. A tree decomposition of a graph G is a tree T with each vertex associated with a set of vertices in G,called a *bag* [11]. Also, the set of bags containing a vertex in G forms a connected component in T. It heuristically computes a tree decompositions and stores shortest-distance matrices for each bag. It is not hard to compute distances from this information. The smaller the size of the largest bag is, the more efficient this method is. Because of the core fringe structure of the networks [12,13], these networks can be decomposed into one big bag and many small bags, and the size of the largest bag is moderate though not small.

✓ Approximate Methods

To gain more scalability than these exact methods, approximate methods, which do not always answer correct distances, also have been studied. The major approach is the *landmark-based approach*. The basic idea of these methods is to select a subset L of vertices as landmarks, and pre compute the distance dG(ℓ, u) between each landmark ℓ ∈L and all the vertices u ∈V . When the distance between two vertices u and v is queried, we answer the minimum dG(u, ℓ) + dG(ℓ, v) over landmarks ℓ ∈L as an estimate. Generally, the precision for each query depends on whether actual shortest paths pass nearby the landmarks. Therefore, by selecting central vertices as landmarks, the accuracy of estimates becomes much better than selecting landmarks randomly [14, 15]. However, for close pairs, the precision is still much worse than the average, since lengths of shortest paths between them are small and they are unlikely to pass nearby the landmarks [4]. To further improve the accuracy, several techniques were proposed They typically store shortest-path trees rooted at the landmarks instead of just storing distances from the landmarks. To answer queries, they extract paths from the shortest-path trees

as candidates of shortest-paths, and improve them by finding loops or shortcuts.

III. RESULTS AND DISCUSSION

✓ A Basic Graph Algorithms

The combination of NE join and EN join can solve a wide range of graph problems in SGC. In this section, we introduce some basic graph algorithms, including Page Rank, breadth first search, and graph keyword search, in which the number of rounds is determined by a user given parameter or a graph factor which is small and can be considered as a constant. We will introduce more complex algorithms that need logarithmic rounds in the worst case in the next sections, including connected component and minimum spanning forest computation.

✓ Breadth First Search

Breadth First Search (BFS) is a fundamental graph operation. Given an undirected graph G(V,E), and a source node s, a BFS computes for every node v ∈V the shortest distance (i.e., the minimum number of hops) from s to v in G. The BFS algorithm in SGC is shown in Algorithm 1. Given a node table V (id), an edge table E(id1, id2), and a source node s, the algorithm computes for each node v the shortest distance d(v) from s to v. Initially, a node table Vd is created with d(s) = 0 and d(v) = φ for v≠ s (line 1). Next, the algorithm iterativelycomputes the nodes with d(v) = i from nodes with d(v) = i − 1. Each iterationi is processed using an NE join followed by an EN join. The NE join propagates d(u) into each edge (u, v) using Vd ✶ NE id,id1 E and produces a new table Ed. The EN join updates all d(v) based on the following rule: (Distance Update Rule): In the i-th iteration of BFS, a node v is assigned d(v) = i iff in the (i−1)-th iteration, d(v) = φ and there exists a neighboru of v such that d(u)≠ φ.The rule can be easily implemented using Vd ✶ EN id,id2 Ed (line 4) in which it also computes a counter nnew which is the number of nodes with d(v) = i. When n new = 0, the algorithm terminates. It is easy to see that the number of iterations for Algorithm 2 is no larger than the diameter of the graph G. Thus the algorithm belongs to SGC if the diameter of the graph is small.

Algorithm 1BFS(V (id),E(id1, id2, s)
1: Vd← !id,id=s?0:φ→d(V)
2: for i = 1to +∞ do
3: Ed ←!id1,id2,d→d1(Vd ✶ NE id,id1E);
4: Vd←!id,((d=φ∧min(d1)≠φ)?i:d)→d(Vd ✶ EN id,id2Ed) | C(d =i) → nnew;
5: if nnew = 0then break;
6:returnVd;

✓ Graph Keyword Search

We now investigate a more complex algorithm, namely, keyword search in an undirected graph G(V,E). Suppose for each v ∈ V, t(v) is the text information included in v. Given a keyword query with a set of l keywords Q = {k1, k2, • • • ,kl}, a keyword search [16, 17] finds a set of rooted trees in the form of (r, {(p1, d(r, p1)), (p2, d(r, p2)), • • • , (pl, d(r, pl))}), where r is the root node, pi is a node that contains keyword ki in t(pi), and d(r, pi) is the shortest distance from r to pi in G for 1 ≤ i ≤ l. Each answer is uniquely determined by its root node r. rmax is the maximum distance allowed from the root node to a keyword nodein an answer, i.e., d(r, pi) ≤ r max for 1 ≤ i ≤ l. Graph keyword search can be solved in SGC. The algorithm isshown in Algorithm 3. Given a node table V (id, t), an edge table E(id1, id2), a keyword query {k1, k2, • • • , kl}, and r max, the algorithm first initializes a table Vr, where in each node v, for every ki, a pair (pi, di) is generated as (id(v), 0) if ki is contained in v.t, and (φ, φ) otherwise (line 1). Then the algorithm iteratively propagates the keyword information from each node to its neighbor nodes using rmax iterations. In each iteration, the keyword information for each node is first propagated into its adjacent edges using NE join, and then the information on edges is grouped into nodes to update the keyword information on each node using EN join. Specifically, the NE join generates a new edge table Er, in which each edge (u, v) is embedded with keyword information (p1(u), d1(u)), • • • , (pl(u), dl(u)) retrieved from node u using Vr ✶ NE id,id1 E (line 3). In the EN join Vr ✶ EN id,id2 E (line 4), each node updates its nearest node pi that contains keyword ki using an amin function, which is defined as: amin({(p1, d1), • • • , (pk, dk)}) = (pi, di)|(di = min1≤j≤kdj) (6) amin is a decomposable since for any two sets s1 and s2 with s1 ∩ s2 = ∅, the following equation holds: amin(s1 ∪ s2) = amin({amin(s1), amin(s2)}) (7) After rmax iterations, for all nodes v in

Vr, its nearest node pi that contains keyword ki($1 \leq i \leq l$) with distance di = d(v, pi) ≤ rmax is computed. The algorithm returns the nodes with di≠ φ for all $1 \leq i \leq l$ as the final set of answers (line 5).

Algorithm 2CC(V (id),E(id1, id2))

1: Vm← !id,min(id,id2)→p(V ✶EN id,id1 E);
2: Vc← !V.id,V.p,cnt(V ′.id)→c((Vm→ V) ✶EN id,p (Vm→ V ′));
3:Vp←! id,((c=0∧id=p)?min(id2):p)→p(Vc ✶ EN id,id1 E);
4: while true do
5: Vs← star(Vp);
6: Eh ← ! id1,id2,p→p1 (Vs ✶NE id,id1 E);
7: V′h← !id,p,min(p,p1)→pm (σs=1(Vs ✶EN id,id2 Eh));
8: Vh←!Vs.id,(cnt(pm)=0?Vs.p:min(pm))→p(Vs ✶ EN id,pV′h);
9: Vs← star(Vh);
10: Eu←!id1,id2,p→p1 (Vs ✶NE id,id1 E);
11: V′u←! id,p,min(p1|p1≠p)→pm (σs=1(Vs ✶ EN id,id2 Eu));
12: Vu ←!Vs.id,(cnt(pm)=0?Vs.p:min(pm))→p(Vs ✶ EN id,pV′u);
13: Vp←!V′.id,V.p((Vu → V) ✶NE id,p (Vu → V ′)) | C(V ′.p≠V.p) → ns;
14: if ns = 0then break;
15: return Vp;
16: Procedure star(Vp)
17: Vg ←!V ′.id,V′.p,V.p→g,(V.p=V ′.p?1:0)→s ((Vp→ V) ✶NE id,p (Vp→ V ′));
18: V ′ s ←!V.id,V.p,and(V.s,V′.s)→s ((Vg → V) ✶ EN id,g (Vg → V ′));
19: Vs←!V′.id,V′.p,(V ′.s=0?0:V.s)→s ((V ′ s → V) ✶NE id,p (V ′ s →V ′));
20: return Vs;

✓ Approaches for General Graphs

The classic solution for shortest path and distance queries is Dijkstra's algorithm [7]. It visits nodes in an ascending order of their distances from the source node, and all the nodes that are closer to the source node than the target node are visited. Thus, many techniques are proposed to reduce the search space. Among these,

(1) bi directional Dijkstra search [10] was proposed to search from both source and target nodes; (2) ALT [15] selects a set of nodes (referred to as landmarks), and pre-computes the distances from each node to landmarks, which are used to prune unnecessary nodes during the search process;

(3) An edge labeling method named ARCFLAG [16] cuts graphs into partitions to reduce the search space;

(4) An indexing and query processing scheme named TEDI [12] decomposesa graph into a tree, and uses the tree index to process shortest path queries; and (5) a 2-hop labeling based exact algorithm was proposed in [19] to deal with large networks.

✓ Approaches for Road Networks

A lot of work focuses on processing shortest path and distance queries on road networks. (1) Different from general graphs, the shortest pathson road networks are often spatially coherent. Path oracles have been proposed for spatial networks [13]. Transit Node Routing (TNR) [18] is a fast and exact distance oracle for road networks. Both of them utilize the property of spatial coherence, i.e., spatial positions of both source and destination nodes and the shortest paths between them that facilitate the aggregation of source and destination nodes into groups sharing common nodes or edges on the shortest paths between them. (2) Road networks are also often assumed to be planar graphs with non-negetive weights, and the properties of planar graphs are further utilized to simplify the search process [20], [18], [17], [15]. Moreover, (3) a (spatial) hierarchical index structure is used by several techniques [11], [14], [17]. For example, Sanders and Schultes [14] proposed a route planning method named Highway Hierarchies (HH) such that only high level edges were considered to compute the path and distance from a source to a far target. Inspired by HH, Geisberger et al. [11] proposed a road network index named Contraction Hierarchies (CH), which is indeed an extreme case of HH. Further, Zhu et al. [17] proposed Arterial Hierarchy (AH) that narrowed the gap between theory and practice in answering shortest path and distance queries. To achieve high efficiency, except for (bidirectional) Dijkstra, all aforementioned approaches require a preprocessing stage to answer a shortest path or distance query. The query answering stage is highly dependent on the preprocessing stage. Thus, the techniques for preprocessing and for query answering are tightly coupled with each other. These techniques are different from ours in two aspects.

(1) These techniques make use of different methodologies. As a result, applying one technique precludes applying another one.However, our routing proxies are a general technique that can be easily combined with other techniques.That is, routing proxies can be used as a pre-processing step before any other technique is applied. Then, any of these technique can be adopted on the reduce graphs, which are typically much smaller than the original graphs, as we have discussed in Section 5. (2) To achieve high efficiency, these techniques usually incur high preprocessing time and space overhead. In contrast, routing proxies are a lightweight technique that scales well to large networks.

Most close to our work is the study of 1-dominator sets in [20]. Different from the fore mentioned techniques, it is proposed for shortest path queries on nearly acyclic directed graphs rather than undirected graphs. When an undirected graph is converted to an equivalent directed graph, each undirected edge is replaced by a pair of inverse directed edges. Hence, 1-dominator sets [20] are not applicable for undirected graphs. However, routing proxies and deterministic routing areas proposed in this study are for undirected graphs, and significantly different from 1-dominator sets (from definitions to analyses to algorithms).

IV. CONCLUSION

We have studied how to speed-up (exact) shortest path and distance queries on large weighted undirected graphs. To do this, we propose a light-weight data reduction technique, a notion of proxies such that each proxy represents a small sub graph, referred to as DRAs. We have shown that proxies and DRAs can be computed efficiently in linear time, and incur only a very small amount of extra space. We have also verified, both analytically and experimentally, that proxiessignificantly reduce graph sizes and improve efficiency of existing methods, such as in directional Dijkstra, ARCFLAG, TNR and AH for shortest path and distance queries. So we are going to implement an new algorithm for finding the shortest path in directed and dynamic graph.

V. REFERENCES

[1] T. Lappas, K. Liu, and E. Terzi, "Finding a team of experts in social networks," in Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2009, pp. 467–476.

[2] M. Potamias, F. Bonchi, C. Castillo, and A. Gionis, "Fast shortest path distance estimation in large networks," in Proc. ACM Int. Conf. Inf. Knowl. Manage., 2009, pp. 867–876.

[3] A. D. Sarma, S. Gollapudi, M. Najork, and R. Panigrahy, "A sketch-based distance oracle for web-scale graphs," in Proc. 3rd ACM Int. Conf. Web Search Data Mining, 2010, pp. 401–410.

[4] E. W. Dijkstra, "A note on two problems in connexion with graphs," NumerischeMathematik, vol. 1, pp. 269–271, 1959.

[5] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick. Reachability and distance queries via 2-hop labels. In SODA, pages 937–946, 2002.

[6] J. Cheng and J. X. Yu. On-line exact shortest distance query processing. In EDBT, pages 481–492, 2009.

[7] I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck. Hierarchical hub labelings for shortest paths. In ESA, pages 24–35. 2012.

[8] R. Jin, N. Ruan, Y. Xiang, and V. Lee. A highway-centric labeling approach for answering distance queries on large sparse graphs. In SIGMOD, pages 445–456, 2012.

[9] F. Wei. Tedi: efficient shortest path query answering on graphs. In SIGMOD, pages 99–110, 2010.

[10] T. Akiba, C. Sommer, and K. Kawarabayashi. Shortest-path queries for complex networks: exploiting low tree-width outside the core. In EDBT, pages 144–155, 2012.

[11] W. Chen, C. Sommer, S.-H. Teng, and Y. Wang. A compact routing scheme and approximate distance oracle for power-law graphs. TALG, 9(1):4:1–26, 2012.

[12] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. Physical Review E, 64(2):026118 1–17, 2001

[13] D. S. Callaway, M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Network robustness and fragility: Percolation on random graphs. Physical Review Letters, 85:5468–5471, 2000.

[14] J. Cheng, Y. Ke, S. Chu, and C. Cheng, "Efficient processing of distance queries in large graphs: A vertex cover approach," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2012, pp. 457–468.
[15] R. H. M€ohring, H. Schilling, B. Sch€utz, D. Wagner, and T. Willhalm, "Partitioning graphs to speedup Dijkstra's algorithm," ACM J. EA, vol. 11, pp. 1–29, 2006.
[16] A. D. Zhu, H. Ma, X. Xiao, S. Luo, Y. Tang, and S. Zhou, "Shortest path and distance queries on road networks: Towards bridging theory and practice," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2013, pp. 857–868.
[17] J. Arz, D. Luxen, and P. Sanders, "Transit node routing reconsidered," in Proc. 12th Int. Symp. Exp. Algorithms, 2013, pp. 55–66.
[18] R. Diestel, Graph Theory. New York, NY, USA: Springer, 2005.
[19] J. E. Hopcroft and R. E. Tarjan, "Efficient algorithms for graph manipulation h(algorithm 447)," Commun. ACM, vol. 16, no. 6, pp. 372–378, 1973.