VOL. 12, NO. 7, APRIL 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

EFFECTIVELY MINIMIZE THE OVERALL TRIP DISTANCE USING CONTINUOUS DETOUR QUERY IN SPATIAL NETWORK

A. Sasi Kumar and G. Suseendran
Department of Information Technology, School of Computing Sciences, Vels University, Chennai, India
E-Mail: askmca@yahoo.com

ABSTRACT

The top-k shortest path discovery is a key process on graphs to determine k-shortest paths between a two nodes with the minimal length. This work precisely holds three processes for ranking the shortest path problem without loop by the way of using top-k shortest path join (TKSPJ) in spatial network. First, Construct transformed graph with side cost by using of input original graph. Second, structural encoding label is used for loop detection and third to find top k shortest path without loop. The main advantage of this work is to reduce the cost and prune the search space. The pre computed shortest paths translating the original graph based on the threshold value has also been introduced, to reduce the search space in a spatial network.

Keywords: graph, shortest path, top-k shortest path, spatial network.

1. INTRODUCTION

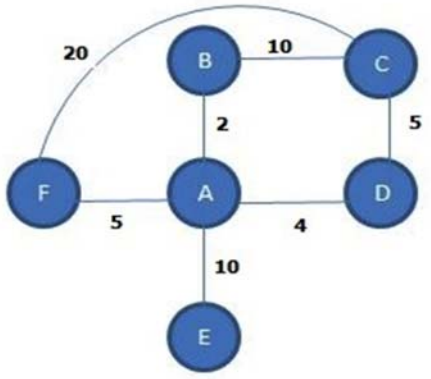
Location based services are used mainly to find the shortest route between the two locations. There may be cases, where the user wishes to find a stopover that will not introduce significant cost to the trip. A pre-computed shortest path to each stop will not produce an overall shortest path. Thus, in order to address this problem, a new query type has been formulated called the detour query, which will use the overall trip distance as the optimization measure. Given a starting and an ending location, the detour query will return a Minimum Detour Object (MDO).

Graph structured data are used in a growing number of applications. A spatial network is a labelled graph whose nodes represent basic and complex geographic entities such as buildings, road segments, routes and spatial groups. The edges represent connections between entities and the labels specify the types of the connections. Edges with the label include an inclusion of an entity in another entity. Directed edge graph with labels are used mainly to identify the start and ending routes in the road segment for spatial network. By using this spatial network/graph as input find the top-k spatial keyword.

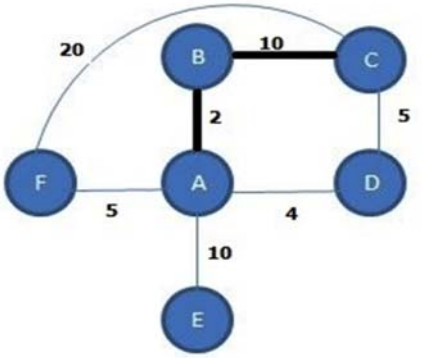
Top-k spatial keyword queries return the k best spatio-textual objects ranked in terms of both spatial proximity to the query location and textual relevance to the query keywords. Euclidean distance [1] restricted to processing top-k spatial keyword queries. In this paper, the interesting and challenging problem of processing top-k spatial keyword queries on road networks. Given a set of spatio-textual objects (e.g., banks annotated with a text). Spatio-textual object consists of two input parameters such as source and destination along with spatial keyword parameter. The output of this work results with two major methods they are 1) shortest path to the query location, and 2) textual relevance to the query keywords.

The top-k shortest paths problem can be classified into two categories [2], the problem of finding the top-k general shortest paths (allowing loops) [3], and the problem of finding the top-k simple shortest paths (without loops) [4, 5]. These two problems face different complexities in graphs. In a positive-weighted graph, the very shortest path between the given pair of nodes is obviously loopless. However, it is possible that the k-th ($k \geq 2$) shortest path has loops. The top-k simple shortest paths problem therefore is significantly harder than the former one due to additional cost for loop detection as well as more search space.

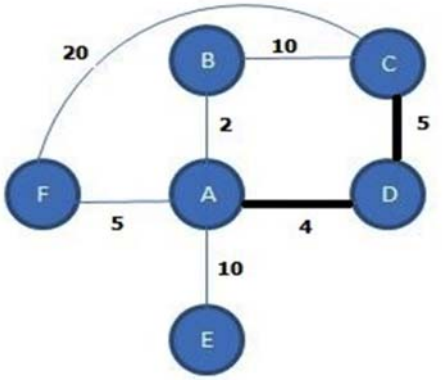
Consider the graph in Figure-1 (a) which represents the nodes along with vertices and edges. The side cost has been assigned for each vertex to travel from source to destination. The shortest path has to be evaluated.



(a)



(b)



(c)

Figure-1. (a, b & c) Sample graphs.

From the Figure-1(b), the start node is A, and the final destination is C. The nodes B and D represent an intermediate stop. A user wishes to go from the start to the end. If the k-shortest path is used, it would return via the path ABC. This is because the shortest path from the starting node A to the intermediate stop is given by AB with a path cost of 2 units. While the other path AD results in a path cost of 4 units and hence it will be rejected. Now, to reach the final destination, it takes the paths BC. The overall cost of this trip is 12 units.

Now, consider the contour detouring method in the Figure-1 (c) that uses the overall path cost as the optimization measure. This method would choose the path AD instead of AB. Then from D, the path chosen will be DC leading to the overall path cost of 9 units. Thus, the overall path cost using contour detouring is less, compared to the k-shortest path. Hence, this is more effective.

Support for realistic location based applications is provided in two aspects. The first aspect is the ability to browse and compare multiple results. Displaying the 'k' Minimum Detour Objects (MDO) at a time will allow the user to browse and select the most satisfying option. The second aspect involves the continuous monitoring of the 'k' MDOs. This monitoring provides users with up to date information. The user may browse information without any current intention to commit to a particular decision. This may cause more time for the users browsing to take decisions, than those who are searching for something specific. A straightforward approach for solving the continuous detour query (CDQ) is evaluating the 'k' MDOs at each intersection along the trajectory. The CDQ solution will incrementally evaluate the 'k' MDOs results at different intersections, according to the usual measure of finding the trip distance. As a result, the repetitive evaluation of the network distances is avoided.

2. MATERIALS AND METHODS

There are several algorithms presently available for solving a K-shortest paths problem without loop in a network in ascending order of their distance between two nodes i.e. Starting and target node.

Every spatial network can be represented as a graph, where all spatial network's nodes and connections are represented as the graph's vertices and edges respectively. Depending on the application this graph may be weighted, directed or un-directed. Thus, any spatial query into the original network can be executed to its corresponding graph representation G. Evidently, the performance of such queries is strongly related to the number of nodes and edges lying into the region, which is a subgraph of G.

Several query processing techniques in spatial networks have been proposed for fundamental query types like window and k-nearest-neighbors queries [6]. To increase the efficiency of these queries various query optimization techniques are used [7]. The classic method for the top-k simple shortest paths is Yen's algorithm [5]. This method first computes the very shortest path from the source node to the target node as the first path. Then, it analyzes each node in the newly discovered shortest path p as the deviation node to generate candidates for the next shortest path using a single-source shortest path discovery. The other shortest path is chosen from all the candidates with the minimal cost. The process continues until k different shortest paths are finally determined. The total time cost of Yen's algorithm is thus O(kn(m+nlogn)), which comes from O(n)single-source shortest path discovery for each of the k shortest paths.

Several attempts have been made to improve the performance of Yen's algorithm. However, the drawback of yen's algorithm is the worst-case complexity cannot be reduced. In order to reduce this complexity Ernesto et al reduced the cost in candidate path generation by discovering the shortest paths incrementally [8]. John Hershberger et al generated the candidate paths with the edge replacement strategy in O(m+nlogn) for each of the k candidates paths. In his work [8], they used the fast replacement algorithm to discover the candidate paths, and switch to a slow but correct method when a loop in the generated path is detected.

Top-k spatial keyword queries on road network are related to keyword queries on relational databases [9] and graphs with external data [10]. However, in relational databases and data graphs, the addressed problem is finding rooted trees of connected vertices that are relevant for the query keywords. There is also related work in the

context of preference queries in road networks and relational databases. Mouratidis *et al* [11] propose processing top-k and skyline queries on road networks assuming additional costs on the edges of the road networks. The results of these queries are used to achieve the execution of top-k queries with minimum cost. Levandoski *et al* [12, 13] discussed a framework for integrating preference queries in database systems. In the context of spatial objects on road networks [14, 15, 16] nearest neighbor queries and range queries was discussed [16] to store the road network and spatial object. One interesting result of this work is the observation that network expansion algorithms present better performance when compared with algorithms based on Euclidean distance heuristics. Lee *et al.* [14, 15] performed using route overlays to improve the performance of nearest neighbor and range queries on road networks.

In the context of spatial keyword queries, Ian de Felipe *et al*. [17] discussed a new data structure that integrates signature files and R-tree. Each node of the R-tree employs a signature to indicate the keywords present in the node sub-tree. Zhang *et al* [18] performed finding the m-closest objects to a given query location that match the set of m query keywords. Cao *et al* [19] introduced finding a group of objects that match the query keywords, minimizing intra-group distance, and the distance among the objects in the group and the query location. These issues are limited to boolean keyword queries and Euclidean distance.

Cong *et al*. [1] and Li *et al.* [20] augmented the nodes of an R-tree with textual indexes such as inverted files. These files are used to prune nodes that cannot contribute with relevant objects. Recently, Rocha-Junior et al. [21] used an indexing structure that associates each term to a different data structure (block or aggregated R-tree) and can process top-k spatial keyword queries more efficiently. Finally, Wu *et al* [22] cover the problem of keeping the result set of traditional spatial keyword queries updated, while the user is moving on a road network, current approaches for processing top-k spatial keyword queries are restricted to Euclidean distance and rely on R-trees to compute the distance between the objects and the query location. Therefore, the techniques proposed cannot be applied in the context of road networks where the distance between the query location and the objects of interest is the shortest path. To overcome all these issues, in this paper proposes a Top-k Shortest Path Join (TKSPJ) method which ranks the top-k spatial keyword and also finds the shortest path then minimize the overall distance in detour path for spatial network

Yen's algorithm [23] is a deviation algorithm that determines only loop less paths. The order of analyzing the nodes in Yen algorithm starts from the deviation node. This gives several changes in the network and provides the proper solution to find the shortest path problem. This loop less path is characterized by its deviation node and its parent node. This yen algorithm is used only for ranking the distance in the spatial network. But, in the proposed work, the distance and the keyword along with shortest path is searched and ranked along with these two attributes.

3. INDEXING AND QUERY PROCESSING IN SPATIAL NETWORKS

3.1 Indexing

In this section, it presents a Top-k Shortest Path Join (TKSPJ) approach has been proposed that indexes the objects lying on the edges of the spatial network based on graph for improving the query processing performance.

3.2 Mapping component

The mapping component [24] depicts a B-tree named map B-tree that maps an edge id to the MBR of the edge. The mapping component also points to the polyline of the edge. The MBR of the edge is used to find the spatio-textual objects lying on the edge through the spatio-textual component.

3.3 Inverted file component

The inverted file component [24] is composed by inverted list's file and vocabulary. The inverted file contains inverted lists identified by a key, composed by the edge id and term id. Each inverted list stores the objects lying on the edge (v, v′) that have a term 't' in their description. For each object, the inverted list stores as follows. Firstly, the network distance between the object and the reference vertex of the edge (e.g., $|v, p_i|$) is stored. Secondly, the impact of the term 't_i' in the description of the object (e.g., $\lambda t_i, p_i$) are stored together in the inverted files to improve the efficiency. The vocabulary file stores the general information about each term t, such as the document frequency of each term. This information is used to compute the textual relevance of the object for a given query.

3.4 Query processing in spatial network

The basic query processing algorithm expands the adjacencies of a query location similarly to Dijkstra's algorithm [25]. The k best spatio-textual objects are maintained in a heap in decreasing order of score. The algorithm stops when the remaining objects cannot have a better score than the score of the k^{th}object already found, or the entire network has been expanded.

The enhanced query processing algorithm [24] performs well when the network is populated, the query keywords that occur frequently, or the query preference parameter gives more weight to the network distance. In these cases, 'k' objects with good scores are found rapidly, which permits the algorithm to terminate earlier. On the other hand, it can perform poorly if the 'k' objects cannot be found rapidly, which can be common in top-k spatial keyword queries on road networks.

4. MINIMIZATION OF OVERALL TRIP DISTANCE USING CONTINUOUS DETOUR QUERY IN SPATIAL NETWORK

4.1 System overview

The framework of the top-k shortest path join architecture is shown in Figure-2 and it is used to construct k- shortest paths from the source node to the target node with the transformed graph.

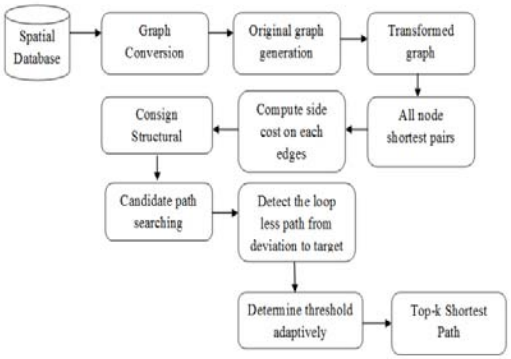


Figure-2. Framework of the proposed Top-k Shortest Path Join (TKSPJ) Technique.

The existing k-shortest path tree (KSPT) has overlaps between Short Path Tree (SPT) branches [3]. These branches are overlapped in such a way that each node appears in the tree exactly k-times in k-different branches. The proposed method implements the Top-k Shortest Path Join (TKSPJ), which constructs a transformed graph with side cost. By using the original input graph, where structural encoding labels are used for loop detection. Adaptive determine threshold minimizes the search space and reduces the terminated cost in each candidate path to find top-k shortest path. It uses the transformed graph in the candidate path discovery termination earlier, to exploit a special property of the Continuous Detour Query (CDQ), where data objects are additively weighted based on their distances to the destination.

The proposed work combines the top-k keyword and the shortest path in the spatial network. Thus the main advantage of this work is to reduce the cost and pruning search space. Pre computed shortest paths translating the original graph into a new graph, have also been introduced; based on the threshold value, which reduces the search space in a spatial network.

4.2 Graph preprocessing

To construct the shortest path tree SPT (t,G) using Dijskstra's algorithm [25], every node is traversed to reach the target node. Each node is labelled with the distance and the successor of the node in the shortest path as given below:

a) To find the possible path from every node to the destination in a graph with minimal length for building the sidetrack due to candidate's path generation earlier.

b) Constructing the transformed graph with side cost, can be easily performed on each edge e=(u,v)to v.cost+e.weight− u.cost

c) To encode the shortest path tree SPT(t,G) with the structural labels to care for the discovery of ancestor/descendant relationship between the nodes efficiently.

The interval labels can be assigned to each node in one time traversal of the tree. The interval label on node u has three attributes such as u.pre, u.post and u.parent, where u.pre and u.post are node u's preorder and postorder number respectively with regard to SPT(t,G). In preceding top-k shortest path algorithm, redundant computations amid candidate path generation is allowed. For example in (Figure-3) the first path is 1→ 2 → 3 → 6 and the second 1→ 2 → 4 → 3 → 6. While starting from divergent deviation nodes, they occur to end with the same path 3 → 6 to reduce the redundant computation cost with the computed Shortest Path Tree (SPT). In order to avoid the loops in the discovered path, it utilizes the structural labels on each node to detect the loops.

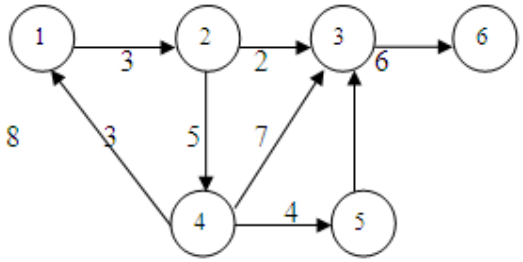


Figure-3. Candidate path termination.

Algorithm: Construct transformed graph

Input: Graph G(v,e) Start node S, Destination node T

Output: Transformed Graph g (v',e') with side cost and Structural encoding labels

1: Find SPT(T,G) every node to same destination with cost;
2: Generate sidecost For Each edge e= (v,e) in G do e.sidecost=v.cost+e.weight-u.cost;
3: Loop detection with three attributes Pre, post order and parent on SPT(t,G) each node
4: Return G;

The advantage of using the transformed graph (Gside) instead of the original graph G is that, the candidate path searching can be concluded generally earlier on the Gside, with the assistance of the encoded pre computed paths. The output of construct transformed graph algorithm is the transformed graph with sidecost.

The attributes of pre order, post order and parent are used to avoid the loop path from start node to target node. The sidecost is also called as side track on each deviation node to avoid traversal speed for discovery of the shortest path.

4.3 Invention of the shortest path encounter

This algorithm is used to discover the shortest path between source and target node from the transformed graph.

Algorithm: Shortest Path Location Encounter

Input: Transformed Graph G(v,e)adj list, Source, Destination

Output: Shortest path from start to target node

1: Initialize: A_1 = shortest-path from source to destination
2: Global ← Local copy of G
3: **for** k = 2 → K **do**
4: **for** i = 1 → [len(A_(k−1)) − 1] d**o**
5: Current Node ← A(k−1) [i]
6: Ri ← Sub-path (root) from source till currentnode in A_(k−1)
7: **for** j = 1 → k − 1 **do**
8: Rj ← Sub-path (root) from source till currentnode in A_j
9: **if** Ri == Rj **then**
10: Next Node ← Aj [i+1]
11: Global (CurrentNode,NextN ode) ← infinity
12: Current Node ← unreachable
13: **end if**
14: **end for**
15: Si ← Shortest-path from current node till destination
16: Bi ← Ri + Si
17: **end for**
18: A_k ← Shortest-path amongst all paths in B
19: Restore original graph: Global ← Local copy of G
20: **end for**

From the given graph Figure-3, this algorithm finds the path for every vertex with the shortest path (i.e. minimum cost) between the source vertex and the intermediate vertex. It can also be used for finding the costs of the shortest paths from a single vertex to a single destination vertex by stopping the algorithm, once the shortest path to the destination vertex has been determined. Dijkstra's algorithm works on the principle that the shortest possible path from the source has to come from one of the shortest paths already discovered. It retrieves the possible shortest path from each vertex and draws the path to move to the destination vertex. Dijkstra's algorithm can be used to find the shortest route between one vertex and all other vertices.

The shortest path encounters algorithm is to produce the shortest path between one node and another, using Dijkstra's algorithm. Another option is to use a heap to keep track of which node should come next, as one property of heaps is that, they always have the next element at the top (either the minimum or the maximum). Since it is possible that in the above algorithm each edge may cause a vertex's position in the heap to change, a heap may require O(|E|log|V|) time. Finally, Dijkstra's algorithm takes O(|E|log|V|) time in this case, as all other terms (such as O(|V|log|V|) for finding the nearest vertex and updating the heap and the O(|V|) initialization) are dominated assuming that there are at least |V| edges in the graph. There are a variety of algorithms for solving the "single-source shortest path" problem for finding the shortest path from a single vertex to all the other possible vertices. Existing algorithms work only in some special cases with less speed. The output of the shortest path encounter algorithm is the shortest path that lies between one node and another node with minimal length.

4.4 K Shortest path discovery

The k shortest path discovery algorithm is to constructs the k shortest path from the source node to the target node in the transformed graph. The top 4 shortest paths P4(p1….p4) from left to rightp1(S, 1, 2, 3, 4, T); p2(S, 1, 2, 3, T); p3(S, 1, 3, 7, T); p4 (S, 1, 3, 5, 6, T) have been shown in Figure-5.

Algorithm: K Shortest Path Location Encounter

Input: Given a preferred path P from Data set D, Graph G (Node N, Edges E), Source, Target

Output: K Shortest path from start to target node

```
function ksp(Graph, origin, sink, K):
A[0] = Dijkstra(Graph, origin, sink);
B = [];
for k from 1 to K do
for i from 0 to size(A[k − 1]) – 1 do
sconNode = A[k-1].node(i);
rootPath = A[k-1].nodes(0, i);
for each path p in A do
if rootPath == p.nodes(0, i) then
remove p.edge(i, i + 1) from Graph;
sconPath = Dijkstra(Graph,sconNode, sink);
totalPath = rootPath + sconPath;
B.append(totalPath);
restore edges to Graph;
B.sort();
A[k] = B[0];
return A;
```

In the given Figure-4 the source S and the target T has been defined for the graph G. Initially, the graph traversal is done from source to destination and the sidecost value is calculated. For the same source to destination, find the k-shortest path to reduce the sidecost. Therefore, earlier path must be removed and the new path should be generated to reach the destination. It is represented in the form of a tree as shown in Figure-5.

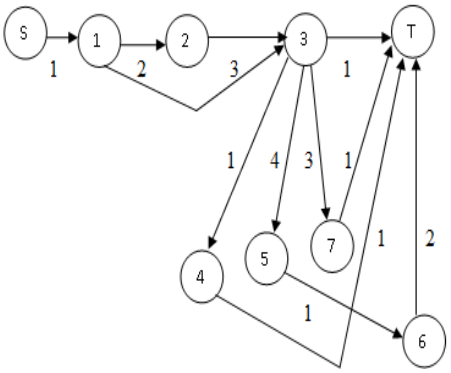


Figure-4. Sample graph of spatial network.

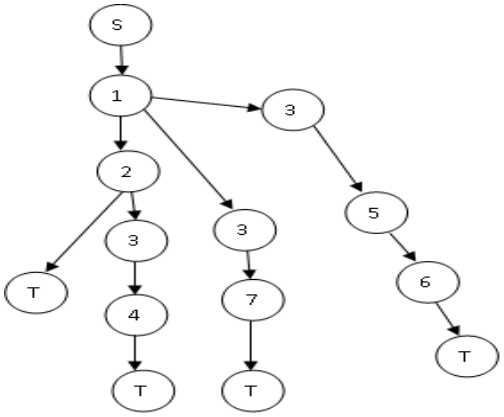


Figure-5. Top-4 Shortest path join (Using Figure-4).

4.5 Proposed Top-k Shortest Path Join (TKSPJ) with detour

Algorithm: Top-k Shortest Path Join with Detour Query
Input: Transformed Graph G'(v',e'), Sidecost and labels, Source, Target
Output: Top-k Shortest path from start to target node
1: Initialize each node v in Gside with v.cost;
2: Sort nodes in ignore Node in term of their pre;
3: Initialize active Q' and insert u (deviation node in Active list);
4: **While** (Q' is not null)
If D is terminating node **then**
Sub path 1= S to D;
Sub path 2= D to T;
5: Ranking the sub paths from S to D and D to T;
6: Return minimal k shortest path;

The top-k shortest path join with detour algorithm is used to discover the 'k' shortest path with the help of a detour to avoid the search space for speeding up the extraction process from one node to any node, which incrementally retrieves data objects and computes node labels as the monitoring process progresses. The computation cost of a kSPT can be greatly reduced by exploiting the fact that the offset assigned to each object 'p' is the distance from 'p' to the destination. Hence, objects that are far away from the destination are likely to be involved in the computation later, than objects closer to the target. Object retrieval is done through monitoring of the labelling distance and incremental retrieval of data objects.

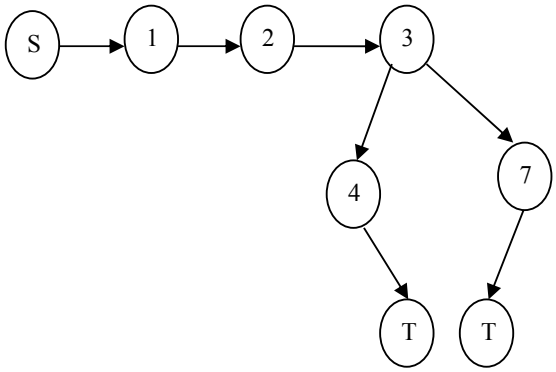


Figure-6. Top-k(2) Shortest path with detour (3).

The Figure-6 demonstrates the shortest path with detour; Firstly find the shortest path from the start to the deviation node and then deviation node assigned as start node for finding the distance between the detour to the target node. The path from S to 3 is fixed after encountering the shortest path and then detects the path from the detour to the target without a loopless path. In top-k shortest path detour algorithm, consider S as the start node, D as the detour node and T declared as the target node. The sub path 1 is denoted as the shortest path from S to D and the sub path 2 from D to T. Finally, detect the overall path from S to T through D, through a loopless path, using structural encoding and side cost value, to avoid traversal of the entire graph.

The output of top-k shortest path detour algorithm is the shortest path with threshold, due to avoiding the whole search from deviation node to target node, for the paths in figure 6. The start to endnode in the network i.e., 3 → 4 and 3 → 7 help in reducing the surplus cost.

5. EXPERIMENTS AND RESULTS

5.1 Statistics of data

In this chapter, real and synthetic data sets are evaluated to find the efficiency and scalability of the TKSPJ method. The proposed method is implemented in the java platform using netbeans 6.1, with the use of yen k shortest path codes, which indicates the extra cost compared with the shortest one. Here, initially the query is processed in the spatial network to find the shortest path distance. For finding the shortest path process, the three methods chosen for comparison are the YEN, KSPT and

the proposed method TKSPJ. These three approaches are compared with various factors to process the shortest path. The three approaches YEN, KSPT and TKSPJ are compared. The three approaches return the same set of top-k objects for a given query. Extensive experiments on both real and synthetic data sets have been conducted, to determine the efficiency and scalability of techniques based on the Table-1 with mentioned dataset, nodes, edges, loop percentage and loop less average.

It employs real and synthetic datasets in the experimental evaluation. All approaches were implemented in the java platform, using netbeans 6.1. In the experiments, the response time (total execution time), index construction time (time to build the index), index size are measured. Table-2 shows the main parameters and values used the experiments.

Table-1. Comparison of k shortest path algorithms.

Methods	Nodes	Edges	Loop	Loop (null)
YEN	15,000	20,000	70%	62%
KSPT	15,000	20,000	80%	74%
TKSPJ	15,000	20,000	35%	23%

Table-2. Parameters evaluated in the experiments.

Parameters	Values
Number of results(k)	10,20,30,40,50
Number of keywords	1,2,3,4,5
Query preference parameter Ω	2,4,6,8,10
Construction parameter Ψ	2,4,6,8,10
Average region cardinality	10,20,30,40,50
Number of layers	1,2,3
Real datasets	India, South Africa, France
Synthetic datasets	R1,R2,R3,R4,S1,S2,S3,S4

Table-3. Parameters of the spatial datasets.

Attributes	India	South Africa	Italy
Total size	104 MB	285 MB	59 MB
Total no.of vertices	50,324	94,124	13,236
Toalno.of edges	59,697	1,32,406	16,759
Avg.no.of lines per edge	7.65	10.34	3.56
Total no.of objects	35,673	52,435	7,456
Avg.no.of objects per edge	0.16	0.07	0.19
Total no.of words	1,56,434	2,12,348	45,231
Total no.of distinct words	13,453	18,321	4,642
Avg.no. of distinct words per object	3.56	4.43	1.53

5.2 Real datasets

It takes the real data sets of three countries namely, India, South Africa and France. In these countries, the map employs the rectangles based on their coordinates (latitude, longitude); Most of the regions in the map describe buildings or a spatial area. Consider the graph with the road network formed by the largest partition of each dataset. Construct the graph of the spatial network and allocate the spatial objects treated as nodes in the road network. Table 3represents some characteristics of each dataset.

5.3 Synthetic datasets

In this, the datasets were obtained by combining the Indian dataset along with road network. It preserves the spatial network structure and the location of the objects in the Indian dataset, to create four datasets, named R1, R2, R3 and R4. From these four datasets, it composes the road network respectively. The synthetic datasets were obtained from the Indian network with various objects. It generates four datasets named S1, S2, S3 and S4 with the values of 300k, 600k, 900k and 1200k.

5.4 Experiments on real datasets

Query processing is performed using the approaches YEN, KSPT and TKSPJ on real datasets.

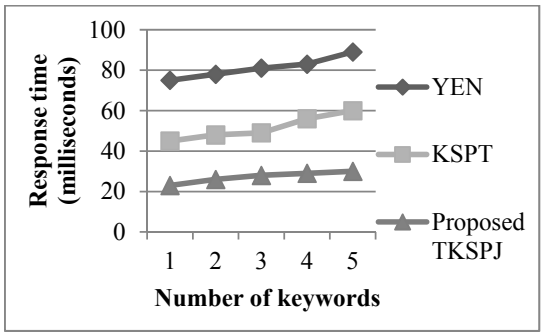


Figure-7. Response time variations with number of keywords.

5.4.1 Varying the number of keywords

Figure-7 presents the response time, while varying the number of keywords in the query. The large number of keywords in the query and the number of objects are relevant. The TKSPJ approach is much better in response time than YEN and KSPT. The number of edges processed by TKSPJ and KSPT is very small for queries with few keywords and increases for queries with more keywords. However, it is much smaller than the number of edges processed by the YEN approach. Note that one single edge of the road network may contain several objects.

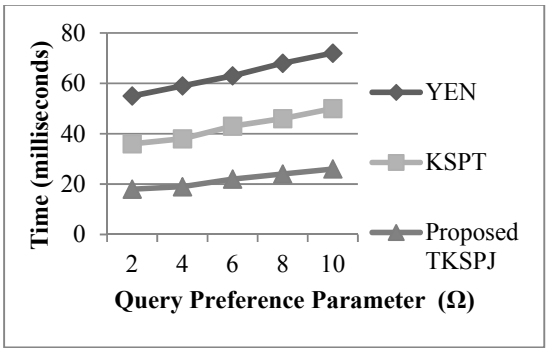


Figure-8. Response time varying the query preference parameter.

5.4.2 Varying the query preference parameter (Ω)

In this study, it evaluates the impact of the query preference parameter as illustrated in Figure-8. The query preference parameter does not present a significant impact on response time.

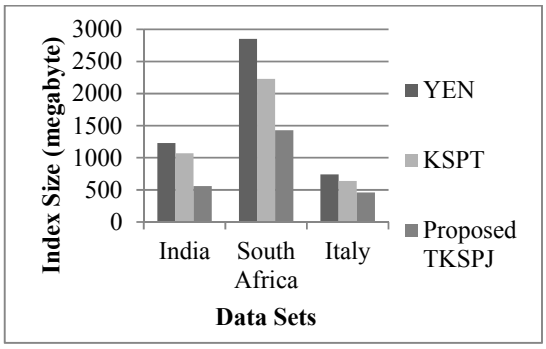


Figure-9(a). Index size varying with datasets.

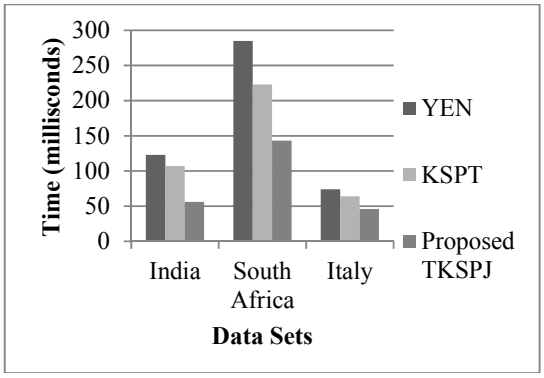


Figure-9(b). Response time varying with datasets

5.4.3 Varying the datasets

In this experiment, it gives the index size and response time for different real datasets, as shown in Figure-9 (a) & (b) represents the index size for the different approaches. The YEN approach retrieves 123 dataset and for KSPT technique 107 is retrieved. Compare to these two methods, the proposed method TKSPJ reduces the index size to 56 as shown in Figure-9(a). The response time varying as shown in Figure-9(b).

5.5 Experiments on synthetic datasets

In this part, it employs synthetic datasets to evaluate the impact of increasing the number of keywords per object and the number of objects (cardinality) on the road network.

5.5.1 Varying the number of keywords per object

Figure-10 shows the response time for varying the number of keywords in the description of the objects (Textual Description Length). The YEN approach is not affected by increasing the description of the objects. The high cost of the YEN approach is in the processing of the edges. Since the number of objects in the datasets does not vary, the number of edges processed is the same for all datasets. However, increasing the description of the objects has an impact on the response time of the KSPT and TKSPJ approaches. The main reason is that it becomes more costly to identify the edges that have relevant objects, since more objects can be textually

relevant for the query. Consequently, more edges have been processed.

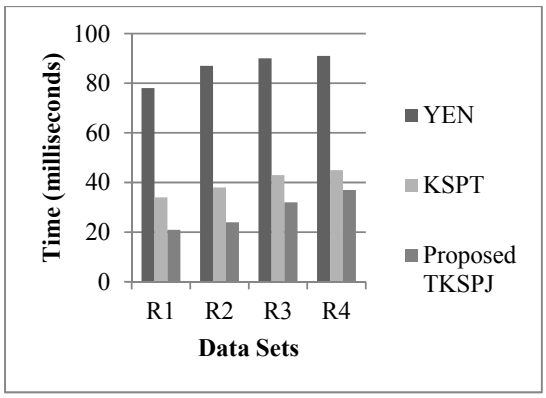


Figure-10. Response time for keyword datasets.

5.5.2 Varying the number of edges expanded

Basically, this work is evaluated with four parameters namely index size, keywords, response time and number of edges expanded (top-k keywords not specified). Therefore, according to the number of edges expanded, the data sets S1, S2, S3, S4 were evaluated and the response time was measured. This is shown in the Figure-11.

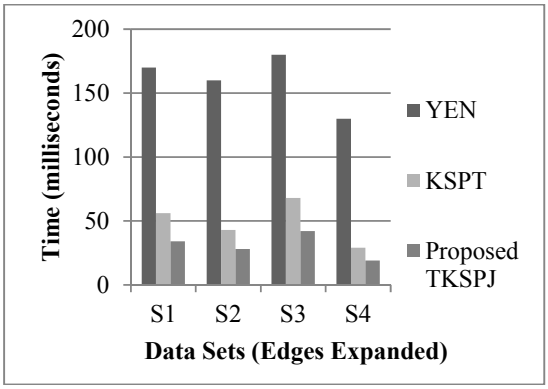


Figure-11. Time analysis for edges expanded in datasets.

Figure-11 shows the number of edges expanded and processed for varying the cardinality of the datasets. Increasing the number of objects has a significant impact on this approach, since it increases the number of edges. In the case of the KSPT and TKSPJ approaches, only the edges considered for top-k will be processed.

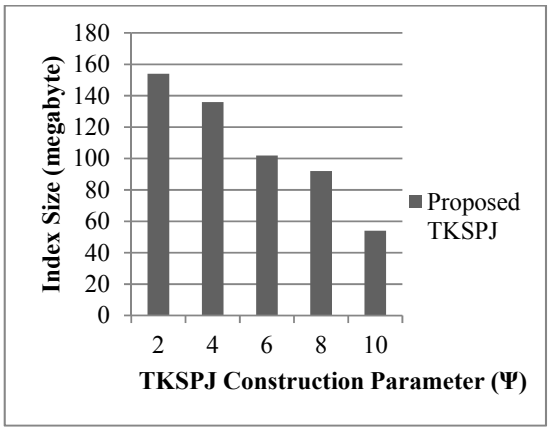


Figure-12(a). Index size varying the KSPJ construction parameter.

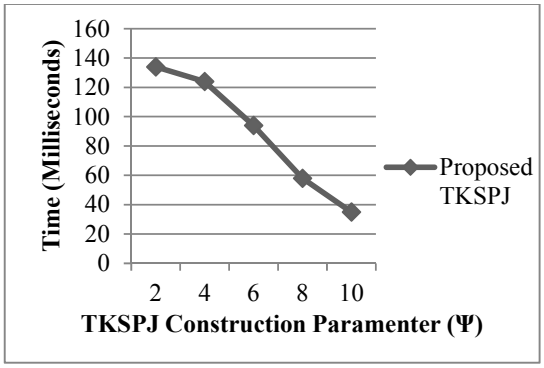


Figure-12(b). Response time varying the TKSPJ construction parameter.

5.5.3 Varying the TKSPJ construction parameter (Ψ)

This work, gives the advantage of employing text similarity in the TKSPJ construction. Figure-12 (a) & (b) show the index size and the response time while varying construction parameter. The textual similarity and grouping of regions are constructed based on the higher value of Ψ using the TKSPJ. Figure-12(a) shows the TKSPJ index size for varying Ψ. The index sizes are varied for small and high values of Ψ. It means incorporating the textual similarity and distance which reduces the index size because the regions created have smaller number of borders. However, only the textual similarity has a positive impact on the response time that is shown in (Figure-12(b)). For small values of Ψ, the index table construction gives more priority to the network proximity instead of the textual similarity among the objects. Therefore, the regions created when Ψis small, containing objects whose text description is dissimilar, also impacts the response time. This experiment demonstrates the advantage of incorporating text similarity during the index construction.

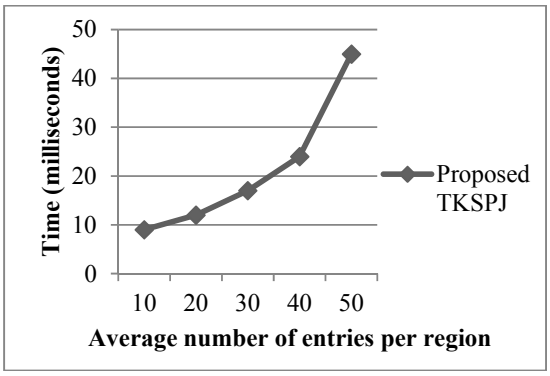


Figure-13. TKSPJ construction time during query processing by varying the average number of entries per region.

5.5.4 Varying the number of entries per region

Figure-13 shows the overlay index construction time and the response time when varying the number of entries per region. If the average number of entries per region is 20, it means that each region at level one has approximately 20 edges; and each region at level two has approximately 20 other regions (around 200 edges). Figure-13 shows that increasing the number of entries per region has a high impact on the index construction time. When the number of entries increases, the vertices also increase. The number of border vertices has a direct impact on the index construction.

5.5.5 Candidate path termination

In this section, evaluation of the output of the proposed method that compared with yen algorithm and KSPT in two ways i.e. loop detection and candidate path termination is performed has been performed. Figure-14 shows the variation of the candidate path termination which has been highlighted. Depending on the number of candidate (intermediate) path, the termination limit increases for the other two existing methods (YEN & KSPT) that is more the number 'k' values the higher the termination limit. In case of the proposed TKSPJ, though the 'k' value increases the termination limit remains reduced thus speeding up the process.

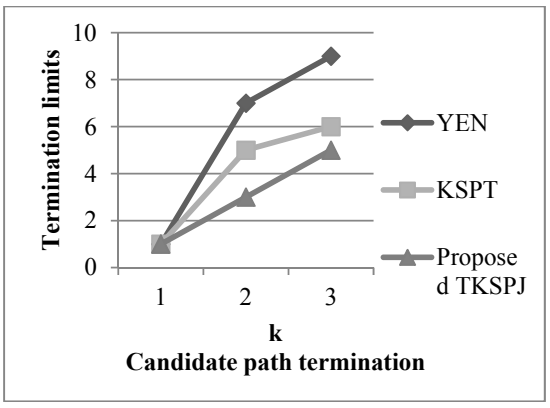


Figure-14. Candidate path termination.

5.5.6 Loop detection

Figure-15 illustrates the loop detection comparison between YEN, KSPT and proposed TKSPJ. The result of the implementation denotes fast discovery of the top-k shortest path, compared to the yen algorithm and KSPT.

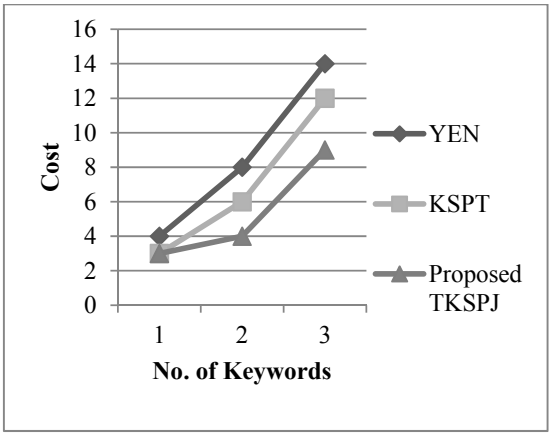


Figure-15. Loop detection.

5.5.7 Graph travel cost

Figure-16 shows the variation of the redundant cost with the three methods. Finally, the proposed method is found to work efficiently to minimize the travel cost and reduce the search time with ranking.

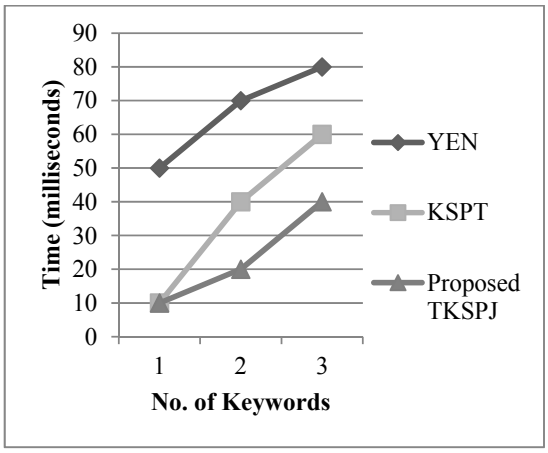


Figure-16. Graph travel cost.

6. CONCLUSIONS

The main goal of the proposed work is to diminish the redundant cost and prune search space in each candidate path generation with an adaptively determined threshold by using the transformed graph/spatial network in the candidate's path discovery termination. These processes are used to speed up the discovery of the shortest path in the non-negative directed graph and to find the candidate paths on the transformed graph more efficiently and diminish the search space with

the threshold. This method provides 93.2 % improvement for finding the shortest path.

7. FUTURE WORK

The top-k shortest path join method is used in the spatial network. It minimizes the search time and reduces the overall travelling cost from the source to the destination. This proposed work is developed for a road network. In future, according to the user specification, it may be developed for any spatial network application. This application can be deployed in the cloud server and cloud will provide a service to the user.

REFERENCES

[1] Cong G., Jensen C.S. and Wu D. 2009. Efficient retrieval of the top-kmost relevant spatial web objects, August 24-28, VLDB Endowment, ACM.

[2] Hershberger J., Suri. S and Bhosle A. 2007. On the difficulty of some shortest path problems, ACM Transactions on Algorithms.

[3] Eppstein D. 1998. Finding the k Shortest Paths, SIAM J. Computing. 28: 652-673.

[4] Hershberger J., Maxel M. and Suri S. 2007. Finding the shortest simple paths: A new algorithm and its implementation, ACM Transactions on Algorithms.

[5] Yen J.Y. 1971. Finding the k shortest loopless paths in a network", Management Science. 17: 712-716.

[6] Sankaranarayanan J., Alborzi H. and Samet H. 2005. Efficient Query Processing on Spatial Networks, Proceedings 13th ACM International Symposium on Geographic Information Systems (GIS), Bremen, Germany. pp. 200-209.

[7] Tiakas E., Papadopoulos A.N., Nanopoulos A and Manolopoulos Y. 2008. Selectivity Estimation in Spatial Network. Proceedings of the ACM symposium on applied computing. 852-856.

[8] De Queiro E., Martins V and Pascoal M.M.B. 2003. A New Implementation of Yen's Ranking Loopless Paths Algorithm, 4OR: A Quarterly J. Operations Research. 1: 121-134.

[9] Park J. and Lee S. 2010. Keyword search in relational databases, Knowledge and Information Systems. 26: 175-193.

[10] Bhavana Bharat Dalvi, MeghanaKshirsagar and Sudarshan S. 2008. Keyword search on external memory data graphs, August 24-30, VLDB Endowment, ACM.

[11] Mouratidis K., Lin., Y and Yiu. M.L. 2010. Preference queries in large multi-cost transportation networks, ICDE, March 1-6. pp. 533-544.

[12] Levandoski J.J, Khalefa M.E and Mokbel M.F. 2011. The CareDB context and preference-aware database system, September 2, VLDB Endowment, ACM, Seattle, Washington, USA.

[13] Levandoski J.J, Mokbel M.F. and Khalefa M.E. 2010. Flexpref: A framework for extensible preference evaluation in database systems, ICDE, March 1-6. pp. 828-839.

[14] Ken C.K. Lee., Wang-ChienLee., Baihua Zheng and Yuan Tian. 2012. ROAD: A new spatial object search framework for road networks, IEEE Trans. on Knowledge and Data Engineering. 24: 547-560.

[15] Lee K.C., Lee W. and Zheng B. 2009. Fast object search on road networks, In EDBT, March 24-26, ACM.

[16] Papadias D., Zhang J., Mamoulis N. and Tao Y. 2003. Query processing in spatial network databases, VLDB.

[17] Felipe I.D., Hristidis V and Rishe N. 2008. Keyword search on spatial databases, ICDE.

[18] Zhang D., Chee Y.M., Mondel A., Tung A.K.H. and Kitsuregawa M. 2009 Keyword search in spatial databases: Towards searching by document, ICDE, March 29-April 2. pp. 688-699.

[19] Cao X., Cong G., Jensen C.S. and Ooi B.C. 2011. Collective spatial keyword querying, SIGMOD.

[20] Li Z., Lee K.C., Zheng B., Lee. W.C., Lee D. and Wang X. 2011. IR-tree: An efficient index for geographic document search, IEEE Trans. on Knowledge and Data Engineering. 23: 585-599.

[21] Rocha-Junior J.B., Gkorgkas O., Jonassen. S. and Norvag. K. 2011. Efficient processing of top-k spatial keyword queries, SSTD.

[22] Wu D., Yiu M.L., Jensen C.S. and Cong G. 2011. Efficient continuously moving top-k spatial keyword query processing, ICDE, April 11-16. pp. 541-552.

[23] Ernesto Q.V., Martins and MartoPascoal. 2003. A new Implementation of Yen's Ranking Loopless Paths Algorithm. pp. 121-133.

[24] Joao Rocha-Junior and KjetilNorvag. 2012. Top-k Spatial Keyword Queries on Road Networks, Proc. 15th ACM International Conference on Extending Database Technology. pp. 168-179.

[25] Dijkstra E.W. 1959. A note on two problems in connexion with graphs, Numerische Mathematik.