ASIAN JOURNAL OF INFORMATION TECHNOLOGY
MEDWELL PUBLICATIONS

Reducing the Search Space in Real-Road Shortest Path Finding using Elliptical Pruning With Geographical Databases

[1]P. Ganesh Kumar, [1]S. Karthik, [2]R. Nidhya and [3]R.M. Bhavadharini
[1]*Department of CSE, SNS College of Technology, Coimbatore, 641035 Tamil Nadu, India*
[2]*Department of CSE, Madanapalle Institute of Technology and Science, Madanapalle, 517325 Andhra Pradesh, India*
[3]*Department of CSE, Easwari Engineering College, Chennai, 600089 Tamil Nadu, India*

Key words: Map pruning, shortest path, map queries, Elliptical pruning and geographical databases

Corresponding Author:
P. Ganesh Kumar
Department of CSE, SNS College of Technology, Coimbatore, 641035 Tamil Nadu, India

Page No.: 163-170
Volume: 19, Issue 9, 2020
ISSN: 1682-3915
Asian Journal of Information Technology

Abstract: Real-road shortest-path finding algorithms involve large set of geographical data which include geo-tagged nodes and edges of the road network. Practical shortest path finding algorithms need minimized search-space for performing these computations by keeping these data in the computer's conventional memory. The hierarchical pruning methods for minimizing the search-space to support real-road shortest algorithms are more suitable for connecting multiple cities and towns via highways. This study is to present a new suitable pruning method for intra-city shortest path finding, called elliptical pruning to bring the geo-tagged nodes and edges in to the memory for the subsequent shortest path finding process for the complex city road-transport networks.

INTRODUCTION

The road-transport network consists of set of directly and indirectly connected nodes and edges. Among many transportation networks, road transportation network play a major and inevitable role in day to day life. People travel to office and children travel to school from their residence every day in the morning and back to home in the evening, materials transported between warehouses, factories, project sites, retail shops etc. Vegetables, fruits and groceries transported from villages to various market places located in cities. It is inevitable to avoid road networks for the above said rapid transportation requirements. Most of the cities already grown in population are unplanned and because of the increased traffic conditions in those cities, finding shortest and fuel-efficient path is so important to reduce the travel time and transportation cost. It impacts up to the level of economic status of a country[1,2]. Choosing the right path among all the available paths to traverse from source node to destination node is the process of finding best path. Mostly shortest distance path will be chosen. However, considering different real-road transport scenarios like, traffic conditions on various time including high traffic time (peak hours), medium traffic time (off-peak hours) and normal hours in a day or a specific weak-day and road-conditions, different paths can also be chosen to travel. Geographical database will serve to find the shortest path from source to destination in a complex road network. The complexity lies in Map digitization, the process of converting geographic data either from a hardcopy or a scanned image into vector data by tracing the features. During the digitizing process, features from the traced map or image are captured as coordinates

in point, line or polygon format. With the help of digitization process all the city junctions, connecting points (nodes) and road-ways are captured in geographical databases to help calculating the route from any given source to a destination. The geographic co-ordinates are formed with latitude, longitude and altitude uniquely refers to any location on the earth three dimensionally. Also, the physical roads (edges) connecting those points in the form of a line will be recorded to produce road-network. These edges recorded will always refer two nodes recorded, namely source and destination physically connected by it. These recorded data altogether form a real-road network and stored in a database. For finding shortest-paths between different nodes, these nodes and connecting-edges with the distances should be present altogether in program memory. As a case study, we could get approximately 1400 nodes per square kilometre within Bengaluru city from a digitized road map in the form of a relational database[3, 4]. Map databases are the sources for the geographical information required for road transport planning, routing, vehicle despatching and other fleet management applications. Road transport networks in the countries like India are very complex networks necessitate travellers to switch back and forth between major roads, arterial roads and sub-arterial roads to get a shortest and comfortable connectivity between source and destinations they are travelling. So, number of connecting nodes in such a networks will be a huge. Computation of shortest path by keeping these entire set of nodes and edges in memory is a little cumbersome process. When we need to find shortest path between locations across the city an effective pruning methodology is required to minimize the search space and serve people travelling often for their daily needs.

Literature review: The studies have been done on the shortest path finding methods like Map Partitioning[1], Pruned Highway Labelling, Flat Encoded Path View (FEPV), Hierarchical Encoded Path View (HEPV) and Transit Node Routing. The approach called pruned highway labelling is to find a nearest highway node to connect the destination via. these highways. The hierarchical algorithm for approximating shortest paths[4] extracts a high-level sub network of relatively long links (and their associated nodes) where routing decisions are most crucial. This high-level network partitions the shorter links and their nodes into a set of lower-level sub-networks. It fixes gateways within the high-level network for entering and exiting these sub networks for improving performance. A space consuming FEPV approach for pre-computing shortest path was replaced by an approach HEPV which divides larger map into smaller fragments and organizes them in a hierarchical manner. Transit node routing considers including important nodes near the starting and destination nodes called junctions and uses a very similar approach like pruned highway-labelling to selectively fetch the route[5].

The map partitioning method is to partition the map area into several parts of geographical portions based on the road-connectivity, road classifications, speed rules, driving patterns and traffic intensity at various time-period in a day and filters the records with pre-conditions given. These techniques require additional traffic data and pre-processed road-transport related information. Because this information are primary inputs for the pruning map records required for the shortest path finding algorithms but also required get updated/ calculated time-to-time to yield better and error free results[6].

The pruned highway labelling and other hierarchical methods are well suitable for finding shortest paths connecting cities and states. Intra-city shortest paths are much different from intercity and interstate shortest paths. Intercity and interstate paths are connected via. highways. So, the hierarchical methods produce good results in pruning nodes for finding shortest paths between cities and towns. Finding shortest path within the city destinations mostly involve even small streets in the city. In this case, the Naive pruning methods sometime outperform the semantic and analytical approaches of pruning for intra-city shortest path finding problems, especially for the cities in countries like India[7, 8]. Improving such naïve pruning may further help us to compute shortest paths quickly and efficiently with the minimum memory and other computing resources. The problem we concerned is limited to the requirement of one-to-one shortest-path finding.

Moreover the approaches discussed above involve the pre-computation of road-network data for increasing performance and most of them are hierarchical ones. And moreover these approaches filter the road connectivity based on the road or node related data. Pre-computed values will become void in case of topological changes happen and they need to get re-computed or refreshed. Road networks of cities with high volume of traffic are prone to get such topological changes like switching between one-way and two-way and other road construction works happens during a period. Very particularly the cities in countries like India. Also adding a promising geographical or area filtering will further improve these above said hierarchical shortest path queries[6].

Naive pre-fetching: A naive and simple method to implement this query is collecting all the nodes which lies in the geographical area of a rectangle falls between the latitude and longitudes of source and destination locations. For example, to find shortest path between a location A and B we shall pre-fetch all the nodes which have latitude between latitude of A and latitude of B as well as longitude between longitude of A and longitude of B.

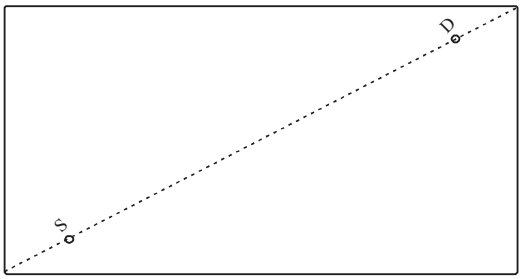

Fig. 1: Pruning with a rectangle

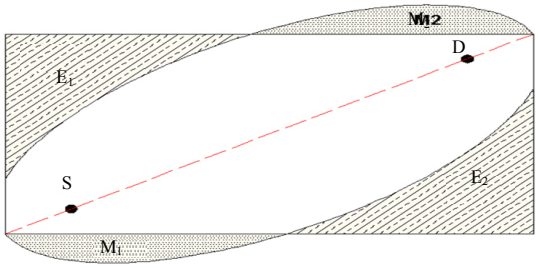

Fig. 2: Excessive selection and missing areas using rectangular area pruning

Fig. 3: Rectangle formed with longer sides parallel to the line connecting source and destination

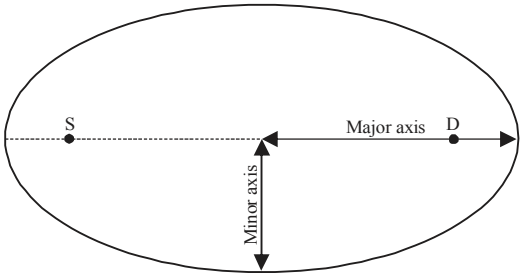

Fig. 4: Ellipse used for pruning

This is clearly a rectangular clipping with the rectangle formed by keeping the source and destination locations at diagonal corners. The rectangular co-ordinates can be found with the permuted values of the latitudes and longitudes of the Source (S) and Destination (D) is shown in Fig. 1. For example, if the (Lat_S, Lon_S) and (Lat_D, Lon_D) are the latitude and longitudes of Source (S) and destination (D), respectively, then the rectangle co-ordinates can be formed with them as (Lat_S, Lon_S), (Lat_S, Lon_D) (Lat_D, Lon_S) and (Lat_D, Lon_D). To avoid missing the roads go towards opposite direction of the destination for some distance from the source location as well as road connectivity that goes beyond the destination for some distance to come back and get connected with the destination location, we may need to expand the rectangle proportionately. The expansion is proportionate scaling to cover little more areas outside the exact source and destination points as found in Fig. 1 Pruning with a rectangle. This expansion can be repeated again in the next attempt if there is a miss in finding shortest path by the algorithm. But the miss can be only known and next attempt will be made with area expansion, after the entire shortest-path finding algorithm run once and fails to identify the shortest path/connectivity between the given source and destination. So, the expansion parameter at the first attempt should be carefully chosen in-order to get a hit.

But the nodes near diagonal points of the rectangle other than the source and destination nodes (the opposite corners) are not favourably going to take part in the resulting shortest-path (in Fig. 2 excessive selection and missing areas using rectangular area pruning), areas marked as E1 and E2) for sure unless otherwise the centre geographical area of the rectangle, we are considering contain water reservoirs or plateaus where roadways are not possible. By including these unwanted nodes in our search space for the shortest-path finding algorithm 8, we demand more memory to keep these nodes and the connecting edges for the computation. It affects the performance of the shortest-path finding algorithm (Fig. 3 and 4).

Another problem is the areas marked as M1 and M2 in the figure are outside the rectangle area but those are near to the source and destination nodes (S and D) and possibly can include nodes to form a shortest path. We are actually missing in this Naive rectangular pruning 7 method.

Sometimes, if the source and destination nodes lies almost in a same axis (Fig. 3 rectangle formed with longer sides parallel to the line connecting source and destination) having either latitude or longitude with small or no differences to each other, the rectangle will be so thin and will miss to get the nodes and edges in the broadened road connectivity. In this special case the pruning method should form a rectangle parallel to the connecting line between the Source (S) and Destination (D) nodes. Forming a rectangle by keeping the sides of it parallel to the connecting line between Source (S) and Destination (D) used to clip like the one in the picture, may not be a feasible solution if the two nodes are far away from each other. The longer path, needs much wider rectangular area to include the curvy deviations of the connecting shortest road-path. So, in this case, the

unnecessarily covered area will be much more. Not only the areas marked as E1 and E2 but it is all the way between the parallel upper and lower boundaries.

MATERIALS AND METHODS

Elliptical pruning method: Here, we are coining a new highly result oriented pruning method for pre-fetching the nodes from map database to find the shortest path between any two nodes given. The method should be as simple as we can include them in a database query. Because, the pre-fetching is going to be done at the geographical database. We are making use of the geographical information to filter-out the unwanted nodes and edges connecting these nodes. The method we hereby call as elliptical pruning since it involves an ellipse which has shortest radius at wide to minimize the number of nodes and can be long enough to go all the way to cover source and destination. The important benefit of using Elliptical pruning is the pre-fetch filtering can be done at database-level to minimize the data transported to the application server.

The Elliptical pruning just checks for the condition to fetch only the nodes present inside the formed ellipse thereby avoids most unwanted nodes and edges to get included in the input of shortest path finding algorithm. The ellipse can be formed with mid-point between source node and the destination node as center and major axis on the line connecting to between Source (S) and Destination (D) nodes (Fig. 5 Ellipse used for pruning). The minor axis can be small enough to promisingly include nodes that form shortest connecting path between source and destination nodes.

Role of minor radius of the ellipse: When a miss occurs that is when the shortest path algorithm fails to find a connecting path from the source to destination, we may need to expand the size of Ellipse to include missed-out road connectivity in that widen area during the previous attempt of the pre-fetching process. So, the pre-fetching process needs careful selection of minor radius to minimize the number of attempts we are going to make as well as to minimize the total area potentially have the road-connectivity. One of the practical solutions we identify for this concern is to have a longer minor radius of the ellipse used for pruning, when there are geographical areas within the ellipse which doesn't have good road connectivity, otherwise shorter minor radius. For decision making we selectively identify the areas where there are no roads possible city-wide and geo-tagging them along with the radius of these road-less areas just to inform the pre-fetching process to increase the minor radius because of the road connectivity going around these areas and not as straight to connect them.

Figure 6 Ellipse with widen minor axis to include areas without road connectivity, shows that there are some road-less areas, here in this example, a lake, inside the ellipse formed for the pre-fetching process. So, the road connectivity goes all around these lake area, resulting in the possible connectivity is wide outside the line connecting Source (S) and the Destination (D). These geo-tagged areas are kept in the database as a separate set of records and may be used to determine the minor axis of the ellipse during the pre-fetching process. This is a one-time process which needs to be done across the city map. These geo-tagged records can be used to determine the length of the minor axis of the pruning Ellipse. If no such area falls near the centre of the Ellipse the minor axis can be minimum otherwise it can be longer than the diameter of the one or more geo-tagged areas between the source and the destination where there are roads not possible. Because of this increment in minor axis, the number of resulting nodes and edges of the pre-fetching process also not increases proportionately. The reason is the geo-tagged road less areas will not have much nodes and edges in it. Just because of that only we are increasing the minor-radius of the pruning ellipse. So, performance of the pre-fetching process will not be getting affected.

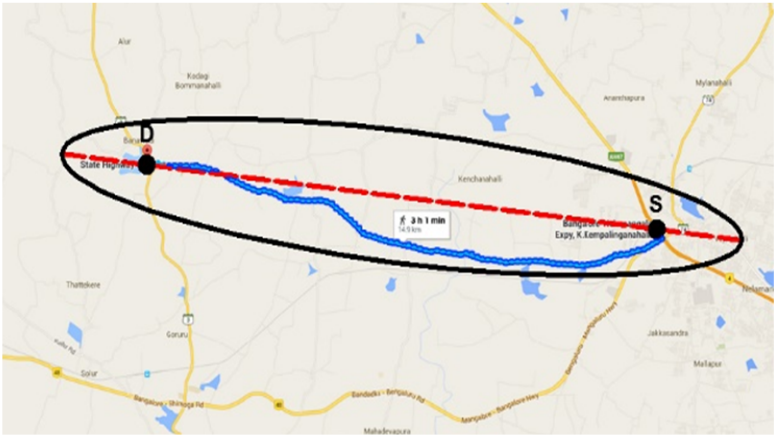

Fig. 5: Elllipse for pruning in a normal city area

Fig. 6: Ellipse with widen minor axis to include areas without road connectively

The practical calculations: Elliptical pruning involves three steps:

- Finding the angle of a line connecting Source (S) and destination nodes directly
- Determining the major and minor axis lengths of the ellipse to be formed
- Apply the filter query in the node table to fetch the nodes lies only in the formed Ellipse

Angle of the line connecting Source (S) and Destination (D) can be found with Eq. 4:

$$Lon_{\Delta} = Lon_{D} - Lon_{S} \quad (1)$$

$$\alpha = \sin(Lon_{\Delta}) \times Cos(Lat_{D}) \quad (2)$$

$$\gamma = \cos(Lat_{S}) \times \sin(Lay_{D}) - \sin(Lat_{S}) \times \cos(Lat_{D}) \quad (3)$$

$$\theta = \text{atan } 2(\alpha, \gamma) \quad (4)$$

Before applying to the equations, all the latitude and longitude values must be converted to radians and finally the angle θ must be converted to degrees. The θ can be the angle of the Ellipse which is centred with the mid-point between the source and destination and major radius will be distance from the mid-point to the source (or destination) node plus the initial expansion value to sufficiently cover the nodes near the Source (S) and Destination (D). So, the mid-point of the Ellipse can be calculated with Eq. 5:

$$[Lat_{C}, Lon_{C}] = \left[\frac{(Lat_{S} + Lat_{D})}{2}, \frac{(Lat_{S} + Lat_{D})}{2}\right] \quad (5)$$

The major radius of the Ellipse can be calculated using Eq. 6:

$$R_{major} = \sqrt{(Lat_{C} - Lat_{S})^{2} + (Lon_{C} - Lon_{S})^{2}} + R_{EXP} \quad (6)$$

According to the observations from our sample set of pruning attempts, initially minor radius of the Ellipse can be one fourth of the major radius. Considering a point (Lat_{P}, Lon_{P}) and the co-ordinate distance (Lat_{D}, Lon_{D}) of it from the centre of the Ellipse (Lat_{C}, Lon_{C}) can be calculated as:

$$[Lat_{D}, Lon_{D}] = \begin{bmatrix} abs(Lat_{C} - Lat_{P}) \\ abs(Lon_{C} - Lon_{P}) \end{bmatrix} \quad (7)$$

To check for a condition that whether the point (Lat_{P}, Lon_{P}) lies inside of the ellipse or not, we need to calculate the following:

$$P_{1} = \frac{(Lon_{D} \times \cos(\theta) + Lat_{D} \times \sin(\theta))^{2}}{(R_{major})^{2}} \quad (8)$$

$$P_{2} = \frac{(Lot_{D} \times \cos(\theta) + Lat_{D} \times \cos(\theta))^{2}}{(R_{major})^{2}} \quad (9)$$

$$P_{1} + P_{2} \leq 1 \quad (10)$$

We select all the nodes that satisfy the condition specified in Eq. 10. The resulting set will be used to find the shortest path between the Source (S) and Destination (D) nodes given.

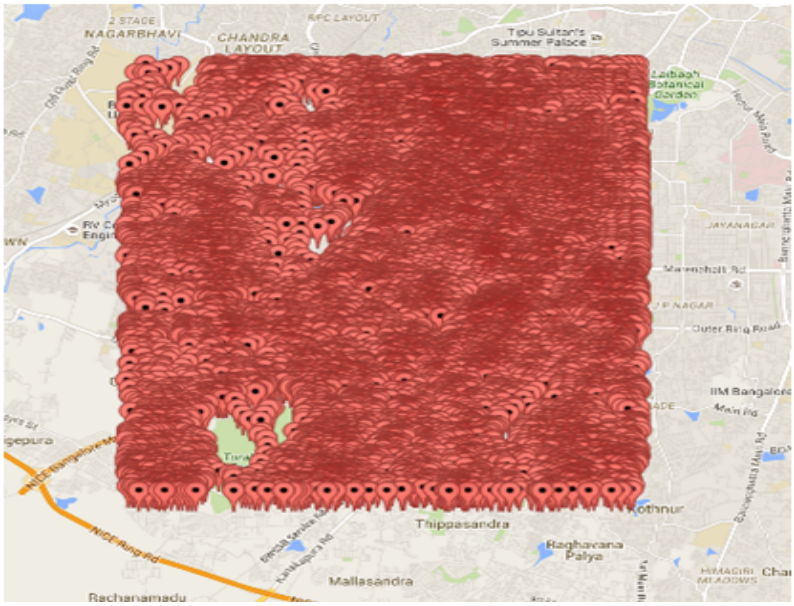

Fig. 7: Pinned resulting nodes found using Naive rectangular pruning

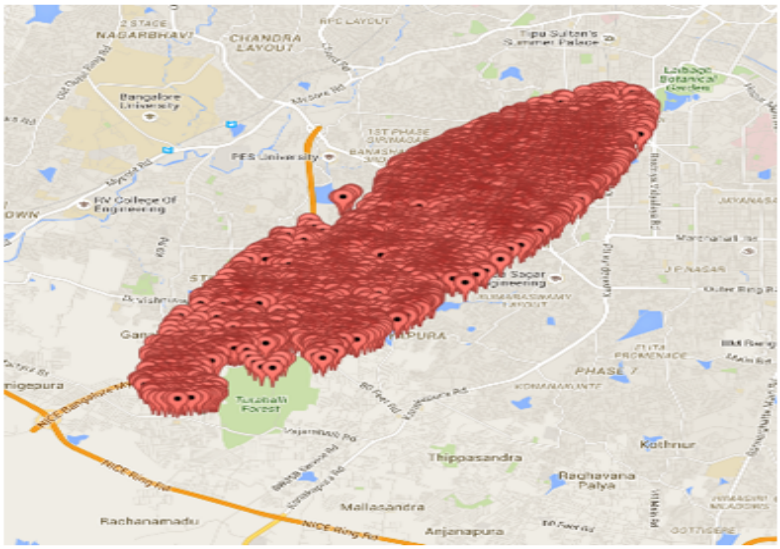

Fig. 8: Pined resulting nodes found using found Elliptical pruning

RESULTS AND DISCUSSION

The analysis of output and performance of Naive pruning and the elliptical pruning methods were done with a sample geographical database with 1,63,584 nodes and 4,41,800 road segments (edges). The other methods like map partitioning and pruned highway labelling are well suitable for intercity shortest paths. They give priority to the other parameters than the distance first. Shortest paths produced from the Naive methods will include all types of road classifications like arterial, sub-arterial, major, minor and even connecting streets to find a minimum distance path connecting two locations. So, the test conducted with 6 sample sets of source and destination nodes for pre-fetching the nodes and edges for computing shortest paths only with Naive and our elliptical pruning. The results were found to be very impressive (Fig. 7-12).

Table 1 shows number of nodes, edges along with the time taken for the queries with the above results the trends are clearly saying that though the elliptical pruning

Table 1: Number of nodes, edges along with the time taken for the queries

	No. of nodes		No. of edges		Time taken for nodes		Time taken for edges	
Tests	Naive	Elliptical	Naive	Elliptical	Naive	Elliptical	Naive	Elliptical
1	1104	819	3226	2123	1587	2023	1686	2039
2	4500	3620	14521	9684	1608	2046	1702	2152
3	9839	6320	34551	20028	1634	2092	1794	2275
4	13693	8505	41643	23643	1664	2153	1817	2280
5	15352	10240	49539	26768	1688	2160	1858	2343
6	17923	11860	58084	33603	1703	2215	1904	2371

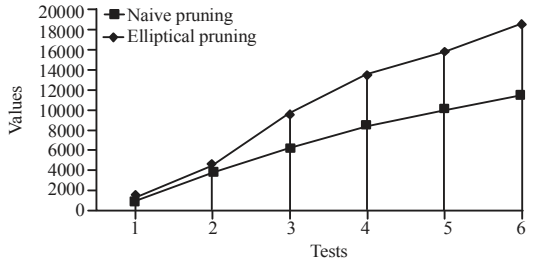

Fig. 9: Number of nodes found during test runs

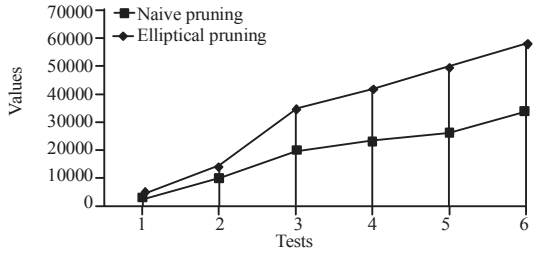

Fig. 10: Number of edges found during test runs

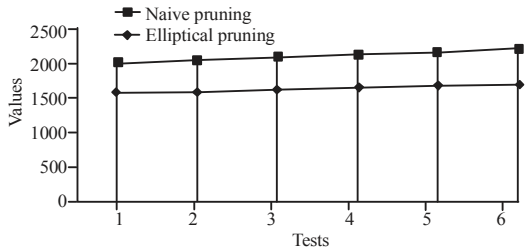

Fig. 11: Time taken for fetching nodes; Querying time for nodes (Milliseconds)

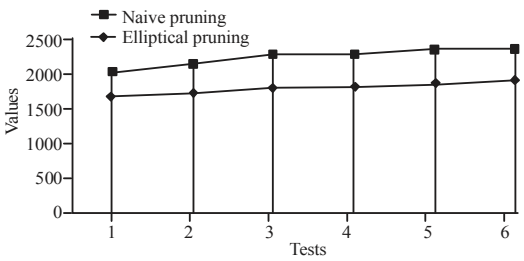

Fig. 12: Time taken for fetching edges; Querying time for edges (Milliseconds)

method takes time to compute and filter-out the unwanted nodes, the numbers of nodes produced to the shortest path algorithm are considerably reduced. The latency time of 1 sec (average) is negligible when compared to the time latency in shortest path computation time with these extra set of nodes and edges. Also, memory requirement for the computation of shortest path will be very less with the pruned set of nodes and edges using elliptical pruning.

CONCLUSION

The Elliptical pruning technique has reduced the generation of redundant nodes and removed the unwanted nodes. The pruning technique has considerably reduced the memory requirement during the computation of the shortest path. The algorithm is useful to find out the shortest path and comfortable path to the destination in complex geographical areas. It serves well for the complex Indian roadways where busy transit is often occurring. Since the elliptical pruning of geographical data for shortest path computation is practically producing minimum set of input for the shortest path computation, this method can be combined with other shortest path querying approaches like pruned highway labelling and map partitioning and Hierarchical Encoded Path View. The hybrid approaches are expected to be a method to further reduce the input data required for shortest path computation.

REFERENCES

01. Gonzalez, H., J. Han, X. Li, M. Myslinska and J.P. Sondag, 2007. Adaptive fastest path computation on a road network: A traffic mining approach. Proceedings of the 33rd International Conference on Very Large Data Bases, September 23-28, 2007, Vienna, Austria, pp: 794-805.
02. Akiba, T., Y. Iwata, K.I. Kawarabayashi and Y. Kawata, 2014. Fast shortest-path distance queries on road networks by pruned highway labeling. Proceedings of the 16th Workshop on Algorithm Engineering and Experiments, January 5, 2014, Society for Industrial and Applied Mathematics Philadelphia, PA, USA., pp: 147-154.

03. Jing, N., Y.W. Huang and E.A. Rundensteiner, 1996. Hierarchical optimization of optimal path finding for transportation applications. Proceedings of the 5th International Conference on Information and Knowledge Management, Nov. 12-16, Rockville, MD, USA., pp: 261-268.
04. Chou, Y.L., H.E. Romeijn and R.L. Smith, 1998. Approximating shortest paths in large-scale networks with an application to intelligent transportation systems. INFORMS J. Comput., 10: 163-179.
05. Bast, H., S. Funke, D. Matijevic, P. Sanders and D. Schultes, 2007. In transit to constant time shortest-path queries in road networks. Proceedings of the Meeting on Algorithm Engineering and Expermiments, January 2007, Society for Industrial and Applied Mathematics, pp: 46-59.
06. Wu, L., X. Xiao, D. Deng, G. Cong, A.D. Zhu and S. Zhou, 2012. Shortest path and distance queries on road networks: An experimental evaluation. Proceedings of the VLDB Endowment, Vol. 5, August 27-31, 2012, Istanbul, Turkey, pp: 406-417.
07. Akiba, T., Y. Iwata and Y. Yoshida, 2014. Dynamic and historical shortest-path distance queries on large evolving networks by pruned landmark labeling. Proceedings of the 23rd International Conference on World Wide Web, April 7-11, 2014, ACM, New York, USA., pp: 237-248.
08. Shehzad, F. and M.A.A. Shah, 2009. Evaluation of shortest paths in road network. Pak. J. Commerce Soc. Sci., 3: 67-79.