Spatial Network k-Nearest Neighbor: A Survey and Future Directives

B. Borhanuddin[1] and B. Solemon[2]

[1]*College of Graduate Studies,*

[2]*College of Computer Science and Information Technology, Universiti Tenaga Nasional,*

43000, Kajang, Selangor, Malaysia.

octavia.sativa@gmail.com

***Abstract*—Nearest neighbor algorithms play many roles in our daily lives. From facial recognition to networking applications, many of these are constantly improved for faster processing time and reliable memory management. There are many types of nearest neighbor algorithms. One of them is called k-nearest neighbor (k-NN), a technique that helps to find number of k closest objects from a user location within a specified range of area. k-NN road network algorithm studies have been through various query performance discussions. Each algorithm is usually judged based on query time over few selected parameters which are; number of k, network density and network size. Many studies have claimed different opinions over their techniques and with many results to prove better query performance than others. However, among these techniques, which k-NN road network algorithm has the highest rate of query performance based on the selected parameters? In this paper, reviews on several k nearest neighbor algorithms were made through series of journal extractions and experimentation in order to identify the algorithm that achieves highest query performance. It was found that with the experimentation method, we can identify not only the algorithm's performance, but also its design flaws and possible future improvement. All methods were tested with some parameters such as varying number of k, road network density and network size. With the results collected, Incremental Expansion Restriction – Pruned Highway Labeling method (IER-PHL) proves to have the best query performance than other methods for most cases.**

***Index Terms*—k-Nearest Neighbor Search; Query Performance; Road Network; Spatial Network Database.**

I. Introduction

Nowadays, there are various map navigation applications that search the best path from one location to another for users. Best path does not necessarily has the least total travel length, but also with least cost, time and avoid inaccessible paths (e.g. traffic jams, road with tolls, road blockages, dead ends, private area, old roads, and so on). The most prominent examples would be Google Maps and Waze applications. The similarity between these applications is that the use of geographic information system (GIS) to manipulate, store, analyze and present geographic data. There are many different technologies involve in the operations of GIS. One of them is the spatial network database (SNDB) for most GIS data such as cities, states, and region divisions of a country. Main features of SNDB are the geometry objects such as edges (lines), polygons and points, and the measurements related with these objects. All these can be manipulated through the spatial network queries facility which involves complex and multidimensional data structure, for example; indexing and join methods. A complex spatial queries in SNDB may use one or more of the basic spatial queries such as *Length*(x), *Area* (x), *Distance* (x, y), and others. One of these advanced spatial queries would be the nearest neighbor (NN) query. NN query returns any closest objects to a query object or point q. The most famous scenario to demonstrate the concept of NN algorithm is the travelling salesperson problem. In this situation, the salesperson needs to travel each node of the city while considering the shortest total distance travelling between them. This problem may be extended to k-nearest neighbor (k-NN) query that finds a set of k closest objects (also known as points of interest (POIs) objects that people normally discovered such as buildings, landmarks, and so on) from q specified. Several studies had been made for k-NN algorithms in SNDB that usually propose the faster (or even the fastest) performance based on query time compared to other methods. Unfortunately, there is no unified 'platform' (e.g. source-code and other related design implementations) for researchers (or even volunteers) to study, modify or share their views on the algorithms.

In this paper, the best k-NN algorithm is demonstrated through both theory and implementation. It is impossible to confirm any best method(s) without experimental replications. Therefore, the main priority is to find shared repository (online) that has SNDB k-NN code examples and compile them into single project. With this project, it gives clue for design flaws and possible improvement. It also confirms which k-NN method(s) has the best query performance aside from sound theoretical it claims.

The rest of this paper is organized as follows: Section II discusses the background and related concepts of NN and k-NN in road network, Section III discusses mainly on the evolution of k-NN theories and methods from 2003 until present, and both Sections IV and V discuss on how the operation of finding the best k-NN algorithms was done and its results respectively.

II. Nearest Neighbor Queries

A. Nearest Neighbor Variant: k-Nearest Neighbor

One of the NN variants and mostly used in SNDB is the k-NN algorithm. The NN query is not necessarily only a single object, but can also be a set of k-NN objects. The variable k is specified as natural numbers which is usually more than zero. For example; if the NN query include k as 5, then the query will return 5 closest objects to q. The closest objects are the POIs objects that people normally encountered such as buildings, landmarks, facilities and so on. There are different definitions of distance as set by users, and therefore the k-NN results would be different. For example, as shown in Figure 1, if user defines a minimum distance between two points, the NN result for *x* would be *y*. Otherwise, the NN of *x* is *z* if maximum distance is defined instead. There are few types of matrix used to find the distance in spatial network. In this example, the Euclidean distances (straight line distances) are used to calculate the length between both points.

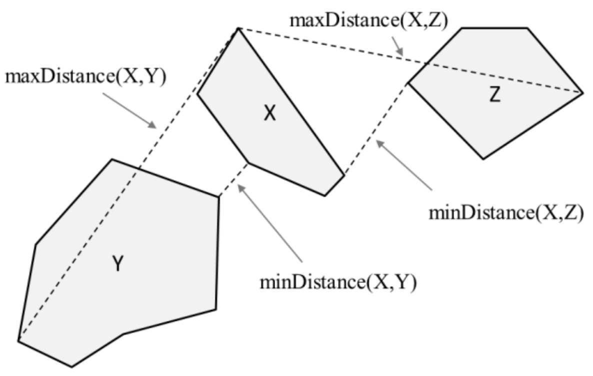

Figure 1: Different Distance Definitions on Nearest Neighbors

B. K-Nearest Neighbor Spatial Queries

k-NN searching methods in SNDB consisted of various sub-algorithms which require the use of Euclidean space distances and specified spatial index. The sub-algorithms used depend on the purpose of the k-NN algorithm itself. One famous example is the Dijkstra's shortest-path algorithm [1]. Some of the most common spatial indices used in SNDB are grid, quadtree, R-tree [2] and etcetera. In this study, both Dijkstra and R-tree methods are frequently discussed in following sections. There are various types of k-NN searching algorithms in road network, and each one has its different purpose when gathering *k* objects whether while traversing the road network or just by scanning parts of regions with Euclidean distances. The k-NN searching methods in this research focused mainly on static objects (query and data points) on spatial network k-NN. The road networks in these situations are usually pre-indexed before query settings can start. The pre-computation or the offline approach involves searching-up data from the pre-indexed data structure. The pre-indexed data are updated only when requested or whenever the datasets are changed.

The R-tree is widely used in k-NN queries, with the branch-and-bound algorithm as the basic way to maintain a list of *k* nearest neighbor candidates in priority queue. The method is based on both B-trees and quadtree, where it is balanced (B-trees) and adjusted groupings of dead-space and dense areas (quadtree) [3]. 'R' is short for Rectangle and some objects reside in these bounding rectangles, or also known as the minimum bounding rectangles (MBRs) because of the minimum distance between *q* and nodes in the tree. There are two types of R-tree branch-and-bound search method; depth-first and best-first. The depth-first search (DFS) visits nodes that are nearer to *q* in every expansion and focuses until leaf node of its predecessor nodes if they are listed as NN candidates. Unlike DFS, the best-first search (BFS) has a priority queue that supports sorted entries based on each minimum distance from *q*. If the node is a leaf, then it will be inserted into NN candidate set and stop if the next popped entry's minimum distance of the priority queue is greater than the last NN candidate's.

As mentioned, shortest-path Dijkstra method is common in k-NN road network algorithms. It consists mainly of two types of set; the shortest-path tree and the unexplored or rejected set [1]. The shortest-path tree keeps track of the visited nodes that are chosen for minimum distance from source node. The last added nodes are compared with the adjacent nodes in subsequent cycles for the sum of distance values from the source. If the total distance is smaller than the other adjacent ones, add that node to the shortest-path tree and update the sum distance. This process will be repeated until all vertices are checked. Dijkstra concept is often used in most k-NN road network traversing process but usually with modified or additional conditions.

III. Related Works

A. Variants of Road Network k-NN Algorithm

There are several types of k-NN methods for road network applications other than the static k-NN which is discussed in next section. One of them is called Continuous k-NN where query objects are mobile while data points may or may not be [4]. The idea here is to let query objects constantly detect any nearest objects on nodes for every expansion. There is also a method that continuously returns a number of nearest objects over a period of time [5]. However, using R-tree in this situation may not be efficient, and therefore a grid-based index is used instead since it is simple and locations of data points can be easily updated. Another type of spatial k-NN algorithm is the Time Dependent *k* Nearest Neighbor (TD-k-NN) search where the mobile query objects are on time-dependent road network [6]. Since to pre-compute all paths with its departure-times requires unreasonable space requirement, the author proposed two types of index: Tight Network Index (TNI) to avoid shortest-path computation and Loose Network Index (LNI) to help identify nearest objects that were undetected by TNI. The Approximate k-NN method is another way to deal with some errors when both data and query points are massive (both static and continuous) to be retained on disk or main memory [7]. Another way for the storage issues would be the Distributed k-NN Processing technique [8] on server-client communication where each moving object is given a 'safe region' (circular or rectangle). If the object leaves the region, it will notify the server its new location to make a new NN query.

B. Spatial Network k-NN Algorithms: The Network Expansion Techniques (2003-2004)

One of the earliest k-NN in road network application defines framework for data models and abstract functionality definitions [9], including other possible algorithms. The detailed data model store road networks and objects' locations while the abstract representation enables for NN querying. The Nearest Neighbor Candidate (NNC) search enables an up-to-date result since the active-query is happened on client to re-compute distances between data points and *q*, while the NNC set is done on server. The Incremental Expansion Restriction (IER) from [10] uses Euclidean distances to find NN objects for every network expansion and adds them into k-NN candidates queue. The last element of this queue is known as lower bound which stores network distance from *q* (d_{Nmax}). The expansion stops when the Euclidean distance of next NN object is greater than the d_{Nmax}. The same author [10] also proposed another method called Incremental Network Expansion that solves the 'false hits' Dijkstra shortest-path algorithm in IER by adding the visited nodes into a sorted expansion queue based on each element's network distance from *q*. For every expansion, any POI found along the segments will be added into k-NN list and stopped if the total network distance from *q* is greater than d_{Nmax} of the last element in k-NN list. The Voronoi Network Diagram method in [11] separates sets of road network into polygons (also known as Voronoi Network Polygons). Each polygon is generated by the data point that covers certain nodes and edges. By pre-calculating distances between boundary points of adjacent polygons from the query point, the total travel distance can then be estimated from these boundary points to internal points of the selected POI's polygon.

C. *Spatial Network k-NN Algorithms: Manipulation of Dijkstra's Shortest-Path Technique (2005-2008)*

Even though the Unique Continuous Search (UNICONS) method [12] concerns more on moving objects, it is also applicable to static query objects. The NNs are pre-computed at selected nodes (also known as condensing points) that store k-NN query result so that the search only happens at these points. Therefore, the collection of POIs along the query path that satisfy query predicate at nodes of this path will be qualified as k-NN result. The next method [13] uses the similar concept of Euclidean distance as in IER. It acts as 'boundary' for each data point or object that covers some vertices or nodes with limited range known as the Island. For each expansion, the algorithm detects any Island that covers the current visited node from *q*. The expansion is repeated until the sum of both network distance from *q* to current visited node and minimum radius of all the Islands is greater than network distance from *q* to the last data point in the priority queue. To reduce priority queue insertions, an index called Spatially Induced Linkage Cognizance (SILC) [14] is used in Distance Browsing (DisBrw) [15] so that nodes that closed to each other can be indexed with region quadtree. Each of the node v_i contains ratio (minimum and maximum) of both Euclidean and network distances between *q* and v_i. Unnecessary NN objects are pruned by the Euclidean distance interval that contains lower (minimum ratio multiply $d_{Euc}(q,v_i)$) and upper (maximum ratio multiply $d_{Euc}(q,v_i)$) bounds on network distance from *q* to v_i. Shortest-path is determined by these intervals and the steps (getting qualified node) are repeated at each quadtree until it reaches object node *t*. Finally, the total interval distance will become the network distance itself between *q* and *t*.

D. *Spatial Network K-NN Algorithms: Combination of Different Components (2009 – Present)*

There is one method which the author in [16] proposed a combination of INE method with mobile agents so that each can execute k-NN query at its node independently. A Central Control System (CCC) is the heart of the system model that controls and dispatches the agents to local area and routing the results to answer queries. The local area agent performs INE search method on specific region with sub-agents on each node to execute k-NN query. If a data point *dp* is one of *q* k-NN candidates and the shortest-path from *dp* to *q* passes through node *v*, then *dp* is considered one of *v* k-NN objects. Another way to improve INE method known as Route Overlay and Association Directory (ROAD) is by skipping nodes without objects by grouping them into regions called Rnets [17]. The Route Overlay Index stores for each node the Rnets it belongs with Association Directory to check whether any given Rnet has any object or not. In each Rnet, every pair of border nodes is defined as shortcut pair. With these shortcut pairs, shortest-path between *q* and v_i may enter any Rnet through one of the border node and exit through the other. The G-tree [18] associates each node with a sub-graph. The partitioned sub-graphs form a tree hierarchy and each non-leaf node stores a set of border nodes and a distance matrix. The distance matrix stores network distances from each child node to other child border nodes. All children nodes that contain objects (POIs) are stored in an occurrence list to prune any empty nodes before traversing. The tree is traversed from *q* (source leaf) and with distance matrices to group shortest-path distances while traversing the hierarchy. If the visited node is an object, then it will be returned as k-NN object. Finally, one author proposed another method to improve IER, which has always been considered the worst k-NN performance, with a shortest-path algorithm called Pruned Highway Labeling (PHL) [19]. The combination (IER-PHL) [20] is similar to IER's k-NN search technique except for the Dijkstra shortest-path. In this case, for every expansion to get network distance between *q* and next NN object, PHL will be used instead. The same author also proposed IER-Gt (combination of IER and G-tree) to take advantages of G-tree's "materialization" property that reduce number of repeated network distance calculation from *q*.

IV. BEST QUERY PERFORMANCE K-NN STUDIES

The methodology to find spatial network k-NN algorithms was designed to avoid unnecessary complexity and repetitive searching. Therefore, only several papers that possess high citation numbers were mostly referred. With these papers, relationships between them and other minor references were made. Theoretical references may not be enough in this case to make conclusion for best technique(s). To find more evidences, paper(s) related to spatial k-NN algorithms that offer any shared online repository were searched and replicated for results.

A. *Finding the Performance Evidence*

The first step is to identify important spatial network k-NN studies which include static query points as part of the experimental evaluations. One objective in mind was to search related papers that contribute to the evolution of SNDB k-NN algorithms in order to understand the underlying problems and reasons for improvement made. By using database search with appropriate filters, it was not difficult to obtain such papers. Few papers were considered as key research [10, 11, 13] since the discussions are related with static objects, indexing and query performance. The challenge was to connect, to find pattern and create relationship between one technique with another since not all papers found have similar perspectives regarding static k-NN query objects. It was found that as more advanced technologies involve in SNDB, the studies on static queries are very limited. One patented paper [21] discusses on relationship between old methods (e.g. IER and INE) and newer ones such as time-based shortest-path algorithms. From this research, although it is not related to static query study, the candidates for comparative purposes have also been identified [10, 15, 17, 18, 20]. It was also found that there were lacked of papers that discussed related experiments according to query performance through sharable platform(s) except [20]. This recent research has made significant contributions of code-sharing repository through GitHub and also discussions on various examples of spatial k-NN results. With the final paper [20], the candidates for best method were confirmed based on its k-NN result and discussion.

B. Reproducing and Testing the Result

With findings based on [20], the second step was to replicate the author's results and study the definitions of best spatial network k-NN query performance. The source code was downloaded from the author's GitHub repository [22], generated and compiled into executable files on local machine. All related datasets were downloaded only for travel distance purposes. Each of the dataset has two types of files; road network edge-weight graph and coordinate files. These datasets were created for the 9th DIMACS (Center for Discrete Mathematics and Theoretical Computer Science) challenge [23] which had been released by US Census Bureau for the point-to-point road graph algorithms. Table 1 shows each dataset information sorted by number of nodes.

Table 1
Road Network Datasets

Code	Description	Node	Edges
DE	Delaware	49,109	60,512
VT	Vermont	97,975	107,558
ME	Maine	194,505	214,921
COL	Colorado	435,666	1,057,066
NW	Northwest USA	1,207,945	2,840,208
CAL	California and Nevada	1,890,815	4,657,742
E	Eastern USA	3,598,623	8,778,114

Some real-world objects' coordinates were obtained from Open Street Map (OSM) as POIs (e.g. fast-food outlets, courthouses, etc.). Because the OSM object sets may not be as precise as in real world, synthetic POIs have been included for more observations. Some random vertices which may simulate real POIs were also selected uniformly for both cities and rural areas. These depend on the object sets' density represented as d (a ratio of number of objects |O| to the number of vertices |V|, |O|/|V|) ranging from 0.0001 to 1. Lower density indicates lesser occurrences objects and vice versa. Value k is varied from 1 to 50 (default value of 10), while d varied from 0.0001 to 1 with default value of 0.001 because the author considered it to match real world object sets density. After all query and random objects were generated, travel distance experiments (varying k, road network density, and road network size) were then run on several k-NN algorithms such as IER, INE, G-tree, IER-PHL, IER-Gt and ROAD. It was found that the results were not much different from [20] even though the local machine has lesser RAM capacity. However, the performance was approximately 5-10% faster due to more advanced processor compared to author's (3.2 GHz Intel i5-4570).

V. EXPERIMENTAL EVALUATION

For the system and hardware settings, the experiment was done in Windows 8.1 operating system (64 bit) on Intel Core i7-4790 at 3.60 GHz. Due to system limitation, it is only allowed up to 16 GB of RAM. It is noted that the author suggests 32GB RAM for large datasets, but it is still possible to perform the experiment by limiting and omitting some datasets to prevent indexes loss. All implementation was done in C++ language with Cygwin command-line interface to create a Unix-like environment and for GNU compilation.

As shown in previous Table 1, Eastern US (E) has the biggest number of nodes and segments. It was found that in current local machine, the PHL road network indexing cannot be constructed for any datasets bigger than E since the main memory is only 16 GB in size. Compared to [20], the author was not able to use PHL with a complete United States road network dataset (23,947,347 vertices, 57,708,624 segments) with 32 GB of main memory. Therefore, the experiment cannot be done further than the E dataset. However, the differences in results can still be seen as network size increases on these four datasets; DE, ME, NW and E.

A. Varying Number of k

For varying *k*, IER-PHL in Figure 2 proves that it significantly performs five times faster than other techniques on NW dataset. This is also confirmed in [20] because IER-PHL manages to collect NN objects as soon as the search expansion starts. Because there are more NN candidates need to be compared when *k* increases, the query time quickly increases between 1 and 10 *k* and slowly increases afterwards when the number of *k* is more than 25. INE is the slowest among all methods because it traverses many vertices and has no materialization facility that can help to avoid false hits during each search cycle. Other method such as IER-Gt, which is the improvement of G-tree, performs better than its former technique in most cases except when the *k* increases and reaches around 20. This is because it needs to visit sub-graphs of next Euclidean NN for each expansion and estimates the distances between *q* and all objects within those sub-graphs. Compared to G-tree, as density increases, it has the advantages of visiting any objects that are within closest sub-graphs (nearer to its border nodes of current tree hierarchy) and therefore has the higher chance to find *k* faster than IER-Gt.

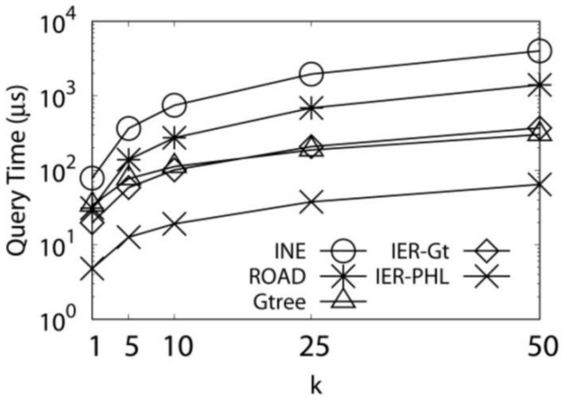

Figure 2: IER-PHL and Other k-NN Methods in Varying *k* (NW, density=0.001)

B. Varying Road Network Density

In Figure 3, *k* is set to 10 and with NW dataset following the default settings for varying density. As density increases, IER-PHL query performance gradually increases, and becomes consistent and stable as it reaches density of 0.1. This pattern is different from other four methods since IER-PHL does not rely on the size of search space in each expansion cycle. It relies on the number of Euclidean NNs found and therefore, as density increases, the chances to false hit unnecessary NNs which are not real k-NN will also increase. However, even if IER-PHL performs slower than

INE, ROAD and G-tree as density increases, it is considered the overall best k-NN method in varying density situations. This is because the degree of query time changes is not extremely affected by increasing road network density values even in bigger road network size. For INE technique, since objects are nearer to each other in higher density, INE is able to visit all nodes nearer to *q* and get to NNs with less visited edges compared to objects that are far from *q*.

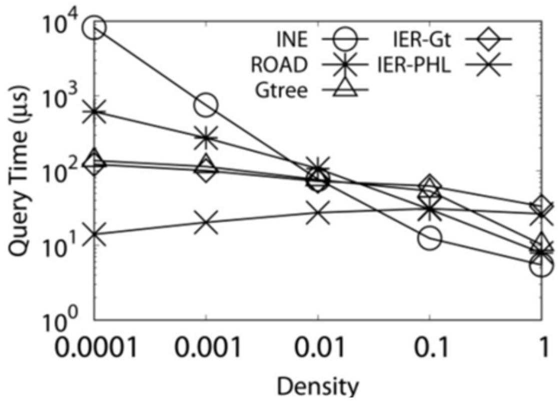

Figure 3: IER-PHL and Other k-NN Methods in Varying density (NW, *k*=10)

However, as network size increases, it performs slower than G-tree and IER-Gt and becomes constant (Figure 4). Both INE and ROAD must track again all vertices which have been de-queued from its expansion (priority) queue. As for G-tree, when density increases, it performs better than IER-PHL and IER-Gt since there is higher chance to obtain more k-NNs towards the end of the tree (source tree node). Although IER-Gt performs slightly better than G-tree in lower density, it becomes slower as density increases since it has to visit multiple sub-graphs with the next Euclidean NN that may lead to false k-NN candidates.

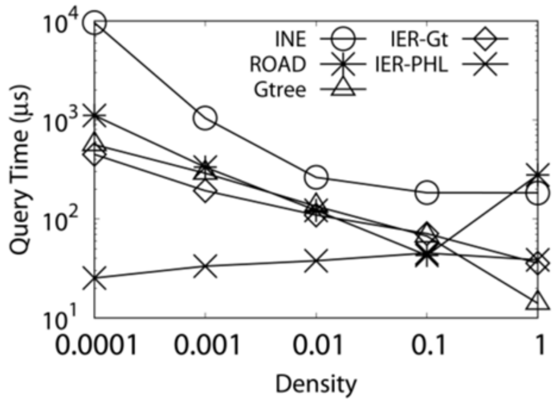

Figure 4: IER-PHL and Other k-NN Methods in Varying density (E, *k*=10)

C. Varying Total Nodes with Default Parameter

The varying number of nodes or vertices reflects the size of the road network. For example, the higher the number of vertices, the bigger the road network dataset would be. There are seven plots for each k-NN method in Figure 5. Each plot corresponds to every dataset shown in Table 1. INE has the slowest performance but remains stable for different number of vertices |*V*|. ROAD has similar pattern to INE except when number of vertices increases in certain stages (when reaching the size of COL and E datasets). ROAD network expansion helps to skip the areas without objects (Rnets) and therefore less expansion cycles.

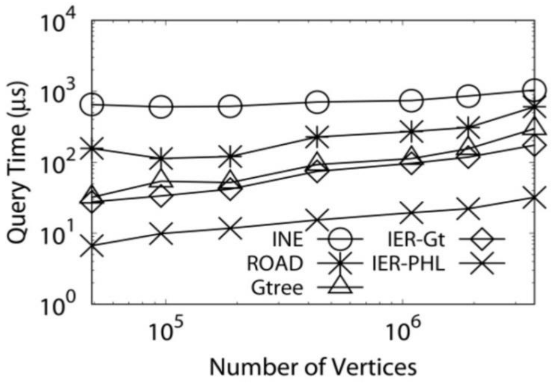

Figure 5: Varying Number of Vertices (density=0.001, *k*=10)

G-tree performs slower as the road network size increases and it seems to have similar pattern to ROAD performance. As number of vertices increases, the nodes at the same depth have more border vertices and the paths to calculate network distance becomes higher in number. For IER-PHL, due to local machine system limitation, PHL in this case can only support up to E road network size. It performs the best among other k-NN methods but suffers the increasing trend of query time as vertices increase just like G-tree and IER-Gt. IER-PHL manages to have gradual and stable increase of query time as road network size increases compared to others (ROAD, G-tree and IER-Gt). As number of vertices increases, so do the NN candidates. Therefore, the longer it takes to find true lower bounds.

VI. CONCLUSION

The replicated results in this research shows that IER-PHL has the best query performance compared to G-tree, INE, ROAD, and IER-Gt. IER-PHL is a combination of two different methods; IER as the one of the oldest k-NN technique, and PHL one of the latest shortest-path algorithm. It performs the best among other recent k-NN techniques in most cases although previously IER was regarded as the worst performing method to find shortest-path by many researchers. k-NN algorithms of SNDB system have been studied by many researchers and each proposed method claimed to have better performance than its predecessor. Unfortunately, there was no shared repository for other researchers to test and compare the actual results unless the author permits to share the work. Having a unified information sharing platform from various k-NN road network algorithms is very important for researchers (or even volunteers) to benchmark their techniques against the best claimed algorithm(s). It is also advantageous for researchers to be aware of new technologies behind SNDB and GIS system and applications for various purposes (e.g. autonomous transportation, logistics, etc.). This research shows that with a shared k-NN project repository, one can study and review various k-NN algorithms' performances. Also, a new feature or k-NN technique can be 'safely' proposed and proved to have better or worse query performance than others.

REFERENCES

[1] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[2] A. Guttman, "R-Trees: a dynamic index structure for spatial searching," in *Proc. of the 1984 ACM SIGMOD Int. Conf. on Management of Data*, New York, 1984, pp. 47-57.

[3] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest neighbor queries," in *Proc. of the 1995 ACM SIGMOD Int. Conf. on Management of Data*, New York, 1995, pp. 71–79.

[4] G. S. Iwerks, H. Samet, and K. Smith, "Continuous k-nearest neighbor queries for continuously moving points with updates," in *Proc. Of the 29th Int. Conf. on Very Large Data Bases*, Berlin, 2003, pp. 512–523.

[5] D. V Kalashnikov, S. Prabhakar, and S. E. Hambrusch, "Main Memory Evaluation of Monitoring Queries Over Moving Objects," *Distributed Parallel Databases*, vol. 15, no. 2, pp. 117–135, Mar. 2004.

[6] U. Demiryurek, F. Banaei-Kashani, and C. Shahabi, "Towards k-nearest neighbor search in time-dependent spatial network databases," in *Databases in Networked Information Systems: 6th International Workshop, DNIS 2010, Aizu-Wakamatsu, Japan, March 29-31, 2010. Proceedings*, S. Kikuchi, S. Sachdeva, and S. Bhalla, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 296–310.

[7] M. Bern, "Approximate closest-point queries in high dimensions," *Information Processing Lett*ers, vol. 45, no. 2, pp. 95–99, Feb. 1993.

[8] Y. Cai, K. A. Hua, and G. Cao, "Processing range-monitoring queries on heterogeneous mobile objects," in *IEEE Int. Conf. Mobile Data Management. Proceedings*. 2004, pp. 27-38.

[9] C. S. Jensen, J. Kolarvr, T. B. Pedersen, and I. Timko, "Nearest neighbor queries in road networks," in *Proc. of the 11th ACM Int. Sym. on Advances in Geographic Information Systems*, New Orleans, 2003, pp. 1–8.

[10] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query processing in spatial network databases," in *Proc. Of the 29th Int. Conf. on Very Large Data Bases*, Berlin, 2003, pp. 802-813.

[11] M. Kolahdouzan ,and C. Shahabi, "Voronoi-based k- nearest neighbor search for spatial network databases," in *Proc. of the Thirtieth Int. Conf. on Very Large Data Bases*, Toronto, 2004, pp. 840–851.

[12] H.-J. Cho and C.-W. Chung, "An efficient and scalable approach to CNN queries in a road network*.,"* in *Proc. of the 31st Int. Conf. on Very Large Data Bases*, Trondheim, 2005, pp. 865–876.

[13] X. Huang, C. S. Jensen, and S. Šaltenis, "The islands approach to nearest neighbor querying in spatial networks," in *Advances in Spatial and Temporal Databases: 9th International Symposium*, SSTD 2005, Angra dos Reis, Brazil, August 22-24, 2005. Proceedings, C. Bauzer Medeiros, M. J. Egenhofer, and E. Bertino, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 73–90.

[14] J. Sankaranarayanan, H. Alborzi, and H. Samet, "Efficient query processing on spatial networks," in *Proc. of 13th Annual ACM Int. Workshop on Geographic Information Systems*, Bremen, 2005, pp. 200–209.

[15] H. Samet, J. Sankaranarayanan, and H. Alborzi, "Scalable network distance browsing in spatial databases," in *Proc. of the 2008 ACM SIGMOD Int. Conf. on Management of Data*, New York, 2008, pp. 43–54.

[16] X. Du, and J. Ji, "Research of K-NN query processing in road networks," in *2nd Int. Conf. on Power Electronics and Intelligent Transportation System (PEITS)*, 2009, pp. 72-76.

[17] K. C. K. Lee, W.-C. Lee, and B. Zheng, "Fast object search on road networks," in *Proc. of the 12th Int. Conf. on Extending Database Technology: Advances in Database Technology*, New York, 2009, pp. 1018-1029.

[18] R. Zhong, G. Li, K.-L. Tan, and L. Zhou, "G-Tree: an efficient index for K-NN search on road networks," in *Proc. of the 22nd ACM Int. Conf. on Information & Knowledge Management*, New York, 2013, pp. 39-48.

[19] T. Akiba, Y. Iwata, K. Kawarabayashi, and Y. Kawata, "Fast shortest-path distance queries on road networks by pruned highway labeling," in *Proc. of the Meeting on Algorithm Engineering and Experiments*, Oregon, 2014, pp. 147-154.

[20] T. Abeywickrama, M. A. Cheema, and D. Taniar, "K-nearest neighbors on road networks : a journey in experimentation and in-memory implementation," in *Proc. of the VLDB Endowment*, 2016, pp. 492–503.

[21] U. Demiryurek, F. Banaei-Kashani, and C. Shahabi, "Efficient k-nearest neighbor search in time-dependent spatial networks," in *Database and Expert Systems Applications: 21st International Conference*, DEXA 2010, Bilbao, Spain, August 30 - September 3, 2010, Proceedings, Part I, P. G. Bringas, A. Hameurlain, and G. Quirchmayr, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 432–449.

[22] T. Abeywickrama, "Road Network k-NN Experimental Evaluation," 2016. [Online]. Available: https://github.com/tenindra/RN-k-NN-Exp. [Accessed: 20-Jun-2006].

[23] "9th DIMACS Implementation Challenge: Shortest Paths.," 2006. [Online]. Available: http://www.dis.uniroma1.it/challenge9/ competition. shtml. [Accessed: 20-Jun-2006].