Enforcing k Nearest Neighbor Query Integrity on Road Networks

Ling Hu[1*] Yinan Jing[2] Wei-Shinn Ku[3] Cyrus Shahabi[4]
[1]Google Inc, Mountain View, CA, USA
[2]School of Computer Science, Fudan University, China
[3]Dept. of Computer Science and Software Engineering, Auburn University, USA
[4]Dept. of Computer Science, University of Southern California, USA
lingb@google.com, shahabi@usc.edu, jingyn@fudan.edu.cn, weishinn@auburn.edu

ABSTRACT

Outsourcing spatial databases, including both road networks and points of interest, to a third party Cloud service provider has attracted much attention from individual and business data owners. With popularity of mobile devices, providing instant and reliable location-based services to smartphones and tablets has been a major means of delivering spatial data to real-world users. Therefore, ensuring spatial query integrity in database outsourcing paradigms is critical. In this paper, we propose a novel road network k-nearest-neighbor query verification technique which utilizes the network Voronoi diagrams and neighbors to prove the integrity of the query result. Unlike previous work that verifies k-nearest-neighbor results in the Euclidean space, our approach verifies both the distances and the shortest paths from the query point to its kNN result on the road network.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Application—*spatial databases and GIS*

General Terms

Algorithms

Keywords

Spatial database outsourcing, Query integrity, Road Networks, Location-based services, Authentication

1. INTRODUCTION

Geospatial technology has advanced significantly over the past decades, especially since the introduction of geospatial technology and location-based services on mobile devices like smartphones. The combination of mobile devices and Cloud-based solutions is creating a versatile ecosystem for reshaping the way geospatial data are stored, managed, served and shared. In this new ecosystem, also known as *database outsourcing*, the data owner (DO) delegates the management and maintenance of its database to a third-party Cloud service provider (SP), and the SP server is responsible for indexing the data, answering client queries, and updating the data on requests from the DOs. Mobile clients, which used to send their queries to the DO, now submit queries to SP and retrieve results from SP directly. The general architecture of the database outsourcing model is shown in Figure 1.

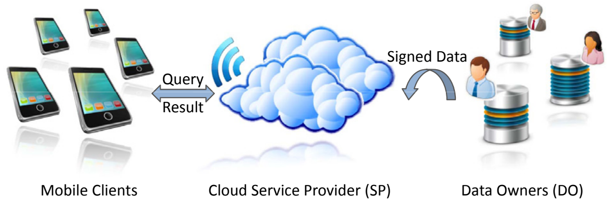


Figure 1: Database outsourcing architecture.

However, as the Cloud service provider (SP) is not the real owner of the data, it might return dishonest or suboptimal results to clients out of its own interests intentionally. For example, an SP which hosts a collection of restaurants might favor some restaurants who pay more advertisement fees. On the other hand, although commercial SPs are generally unlikely to provide unfaithful results to users, they could act malicious involuntarily and serve false results unintentionally to the end users, e.g., when SP is under attack, or while the communication channel is compromised between SP and clients. Therefore, providing a mechanism that allows clients verify the integrity of query results is necessary.

Most existing works [1–3, 5, 6, 9, 11] solve query integrity problems in the Euclidean space where the distance between two objects is measured by a straight line. That is, the distance depends only on the locations of the two points. Unfortunately, Euclidean distance is not an appropriate distance measure for many real-world applications where objects can only move on predefined paths, such as streets on a road network. Furthermore, there might exist multiple paths between two points on a road network. Therefore, simply verifying distances is not sufficient for queries on road networks. Consequently, we need an approach which can verify the path between two points. To the best of our knowledge, the only related work that studies spatial query verification based on road networks is [10]. However, approaches

[*]This work was completed when the author was a PhD student at USC's InfoLab.

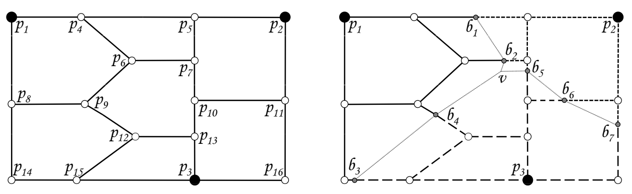
proposed in [10] do not apply to the network k-nearest-neighbor verification problem because their approach only verifies whether the path returned by the SP server is the valid shortest path between two points, but cannot verify whether a point of interest (POI) is the closest point (or the i^{th} closest point) to the query point.

Motivated by the above observations, in this paper we propose a novel query verification approach that can verify network k-nearest-neighbor queries with regards to both distance and path, that is, the k resulting objects have the shortest distances to the query point among all the POIs in the database and the path from the query point to each k-nearest-neighbor result is the valid shortest path on the network. Our approach authenticates network k-nearest-neighbor query results based on the neighborhood information derived from the POI dataset and the underlying road network. Specifically, before transferring its POI database and the road network to the SP server, the owner first partitions the network into disjoint network Voronoi cells each of which is *governed* by a POI object. Next, a signature is created on each POI and its surrounding neighbors to formulate an authenticated object. Finally, all the authenticated objects are transmitted to the SP server which hosts the database service and serves query results to users.

The rest of the paper is organized as follows. Section 2 discusses the network Voronoi diagram and its properties. Section 3 introduces the authentication data structure. Section 4 describes the verification algorithm for network k-nearest-neighbor queries and proves its correctness. Finally, Section 5 concludes the paper and provide directions of future work.

2. PRELIMINARIES

A road network can be modeled as a weighted graph $\mathbb{G}(\mathbb{V}, \mathbb{E}, \mathbb{W})$ consisting of a set of vertices $\mathbb{V} = \{p_1, p_2, \ldots, p_n\}$ and a set of edges $\mathbb{E} = \{e_1, e_2, \ldots, e_k\}$ connecting vertices. $\mathbb{W}$ represents the cost of each edge in $\mathbb{E}$, for example, the distance, the traveling time or the toll fees. Given a set of points of interest (POI) P ($P \subset \mathbb{V}$) which are restricted to the edges on the road network, one can construct the Network Voronoi Diagram (**NVD**) by expanding shortest path trees from each POI simultaneously until the shortest path trees meet. The meeting points are called *border points* with the property that the costs from the border point to the two neighboring POIs are the same.



(a) Road network. (b) Network Voronoi diagram.

Figure 2: An example of road network and network Voronoi diagram.

An example of a road network and a set of POIs are shown in Figure 2(a), where p_1, p_2, and p_3 are points of interest and p_4-p_{16} are intersections on the road network. The corresponding network Voronoi Diagram is shown in Figure 2(b). In a road network, the distance between any two points p and q is measured by the shortest path, denoted as $d_n(p, q)$. For each POI p_i, we define the following:

DEFINITION 1. ***Dominance Region***

$$Dom(p_i, p_j) = \{p | p \in \bigcup_{i=1}^{k} e_i, d_n(p, p_i) \leq d_n(p, p_j), i \neq j\}$$

p_i and p_j are POIs. The dominance region defines all the points on all edges in E that are closer to p_i than to p_j (or have equal distances to p_i and p_j).

DEFINITION 2. ***Network Voronoi Cell***
$V(p_i) = \bigcap_{j \neq i} Dom(p_i, p_j)$

All road segments closer to p_i form the $V(p_i)$, and p_i is called *generator* of the $V(p_i)$. If two generators share one or more border points, they are *Voronoi neighbors.* In Figure 2(b), the network Voronoi cell of p_1, p_2 and p_3 are represented by line segments with different style separated by border points b_1-b_7. Note that network Voronoi cells are mutually exclusive (they do not overlap, except at border points) and collectively exhaustive (every point on the road network belongs to at least one generator). Consequently, the network Voronoi diagram can be defined as follows.

DEFINITION 3. ***Network Voronoi Diagram***
$NVD(P) = \{V(p_1), \ldots, V(p_n)\}$

Given the Voronoi diagram of a set of POIs P and a road network $\mathbb{G}$, the first nearest neighbor of a query point q has the following property [8].

Property 1. *Let $V(p)$ be the Voronoi cell of p on a road network, the nearest neighbor of a query point q is p, if and only if $q \in V(p)$.*

Furthermore, the following property holds for the k nearest neighbors of a query point q on a road network [4].

Property 2. *Given the network Voronoi diagram of P and a query point q on a road network, let $p_1, p_2, \ldots, p_{k-1}$ be the $k-1$ nearest neighbors of q, then the k^{th} nearest neighbor of q is among the Voronoi neighbors of $p_1, p_2, \ldots, p_{k-1}$.*

3. AUTHENTICATION DATA STRUCTURE

Before outsourcing the dataset to the Cloud, DO first computes the network Voronoi cell $V(p)$ for each generator p. Next, Voronoi neighbors of each generator are retrieved and stored in a vector $Nbr(p)$. In Figure 2(b), the Voronoi neighbors of p_1 are p_2 and p_3.

After computing the network Voronoi cells, DO signs each POI p along with the Voronoi cell $V(p)$ and its neighbors $Nbr(p)$ to create an authenticated network Voronoi cell o_{anvc} as follows:

$$o_{anvc} = \langle p, V(p), Nbr(p), \mathbb{S} \rangle$$

$$\mathbb{S} = sign(h(p) | h(V(p)) | h(Nbr(p)))$$

where h is a one-way, collision-resistant hash function and '|' represents the concatenation of two binary strings. Next, DO transmits the authenticated network Voronoi cells, denoted as $\mathbb{O}$, to SP which hosts the database service, builds spatial indexes such as VR-tree [4] for the data, and processes kNN queries on behalf of the DO.

Subsequently, the SP server constructs the verification object ($\mathcal{VO}$) for the kNN query result which contains the POIs, the shortest path to each POI, the network Voronoi cell and the Voronoi neighbors of each POI, together with the signatures generated by the DO. When there are multiple signatures, the SP server employs a signature aggregation technique [7] to condense multiple signatures into one, thus reducing the size of the $\mathcal{VO}$. Finally, the SP server transfers the $\mathcal{VO}$ back to the query client.

4. VERIFYING NN ON ROAD NETWORKS

Generally, a query verification process consists of two sequential steps, that is, signature verification and geometry verification. In the signature verification step, on receiving the result of a kNN query, a client first examines the signature attached to the $\mathcal{VO}$ to ensure that all objects in the result set originated from DO. On receiving an aggregate signature, the client verifies the signature by employing the corresponding aggregate signature verification algorithm [7].

Next, the client verifies the geometry property of the kNN result using the road network kNN verification algorithm shown in Algorithm 1. The inputs to the algorithm are the query point q, the verification object $\mathcal{VO}$, and the parameter k. The verification object $\mathcal{VO}$ contains the authenticated network Voronoi object of every POI in the kNN result, including the generator POI, the Voronoi cell of the generator and its neighbors. The method $\mathcal{VO}.getkNN(i)$ returns the network Voronoi object of the i^{th}NN on line 2 and line 9. H is a min-heap containing all the Voronoi neighbors of already verified NNs and objects in H are sorted by their network distances to the query point q, represented as $h.nd$. $Visited$ is a set which maintains all the POIs that the client has seen so far to avoid examining the same object twice.

Algorithm 1 VerifyNetwork$k$NN(q,$\mathcal{VO}$,k)

```
1.  $H \leftarrow \emptyset$; $Visited \leftarrow \emptyset$; $g \leftarrow \emptyset$;
2.  $(p, V_{link}, Nbrs) \leftarrow \mathcal{VO}.getNN(1)$;
3.  if ($q \notin V_{link}$) then
4.     return false;{the 1$^{st}$ NN fails by Property 1}
5.  end if
6.  $Visited.add(p)$;
7.  $g \leftarrow V_{link}$; $H \leftarrow Nbrs$; $Visited \leftarrow Nbrs$;
8.  for $i = 2$ to $k$ do
9.     $(p, V_{link}, Nbrs) \leftarrow \mathcal{VO}.getNN(i)$;
10.    if $p \notin H$ then
11.       return false;{Property 2}
12.    end if
13.    $g \leftarrow g \cup V_{link}$;
14.    $minDist \leftarrow MaxValue$; $minPt \leftarrow null$;
15.    for all ($h \in H$) do
16.       $h.nd = computeSP(g, q, h.poi)$; {Lemma 3&Lemma 4}
17.       if ($h.nd < minDist$) then
18.          $minDist \leftarrow h.nd$;
19.          $minPt \leftarrow h$;
20.       end if
21.       if ($minDist < p.nd$) then
22.          return false; {$minPt$ is closer to $q$ than $p$ to $q$}
23.       end if
24.    end for
25.    if ($p.nd == minDist$) then
26.       $H.remove(minPt)$;{the $i^{th}$NN is verified}
27.       for all ($nbr \in Nbrs$) do
28.          if ($nbr \notin Visited$) then
29.             $H \leftarrow H \cup nbr$; $Visited \leftarrow Visited \cup nbr$;
30.          end if
31.       end for
32.    else
33.       return false;{the $i^{th}$NN is not verified}
34.    end if
35. end for
36. return true;
```

The network kNN verification algorithm starts with verifying the first NN of the result (lines 3-5). According to Property 1, we know that the query point q must be on a road segment inside the Voronoi cell of its first NN. Therefore, given the first NN p_1 and its Voronoi cell $V(p_1)$, a client checks whether the query point q falls on one of the road segments inside $V(p_1)$. If this is false, p_1 is not the first NN of q and the verification fails. Otherwise, the generator p_1 is the first NN of q. The actual distance and the shortest path from q to the generator p_1 can be verified using the Dijkstra's algorithm on the subgraph inside $V(p_1)$.

Lemma 1. *Let $V(p)$ be the network Voronoi cell of p, the shortest path from any point $q \in V(p)$ to the generator p only goes through road segments within $V(p)$.*

Lemma 1 can be also generalized to the shortest path from a query point to its k^{th} nearest neighbors on a road network.

Lemma 2. *Given the network Voronoi diagram of P and a query point q on a road network, let $p_1, p_2, \ldots, p_{k-1}$ be the $k-1$ nearest neighbors of q and $V(p_1), V(p_2), \ldots, V(p_{k-1})$ be the corresponding network Voronoi cells. the shortest path from q to the k^{th} nearest neighbor p_k only goes through the union of $\{V(p_1), V(p_2), \ldots, V(p_{k-1}), V(p_k)\}$.*

Lemma 1 and Lemma 2 can easily be proved by contradiction. Due to the space limitation, the proof is omitted. Actually, Lemma 1 ensures the shortest path from q to p_1 is fully contained in $V(p_1)$. Therefore, the Voronoi cell $V(p_1)$ is sufficient for computing the shortest path from q to p_1 on the client. As a result, both the distance and the shortest path from query point q to first NN p_1 can be verified.

Next, before verifying the second NN, the road segments V_{link} inside the Voronoi cell of p_1 are merged with g, which is a subgraph of the road networks inside the Voronoi cells of already verified NNs, and all the neighbors of p_1 are inserted into H and $Visited$ set (lines 6-7). Then the subsequent for loop (lines 8-35) iterates through each object in the kNN result set obtained from $\mathcal{VO}$ (line 9).

If the current NN p is not in H (i.e. the current NN returned is not one of the Voronoi neighbors of previously verified NNs), then p is not the i^{th}NN of q (according to Property 2), and the verification fails on line 11. Otherwise, p is one of the neighbors of already verified NNs, and we need to verify that the network distance of p is the smallest among all the Voronoi neighbors (line 13-34). Next, we first verify the second NN by utilizing the Voronoi neighbors $Nbr(p_1)$ of the first NN.

Lemma 3. *Given a query point q, the first NN p_1, the network Voronoi cell $V(p_1)$, and the Voronoi neighbors $Nbr(p_1)$ of p_1, the shortest path distance from q to the second NN of q can be verified.*

Proof: According to Property 2, the second NN must be one of the Voronoi neighbors of the first NN. Therefore, if we know the network distance from the query point to each of the Voronoi neighbors in $Nbr(p_1)$, then we know which generator is the second NN. Given the query point

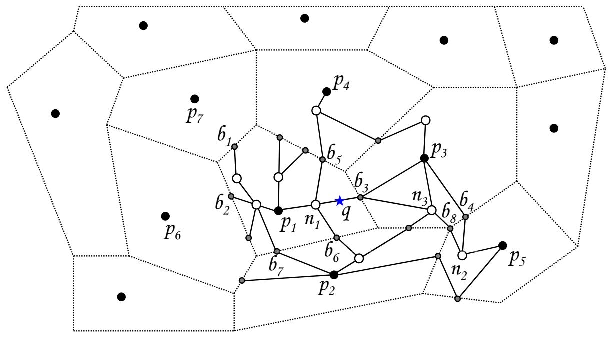


Figure 3: Shortest path on road networks.

q, the first NN p_1, $V(p_1)$, and $Nbr(p_1)$, we can compute the shortest path distance from q to each Voronoi neighbor of p_1 based on the subgraph inside $V(p_1)$. Specifically, we first compute the distances from q to all the border points on $V(p_1)$ using the Dijkstra's algorithm based on the subgraph inside $V(p_1)$. Next, because the network distances from a border point to the adjacent generators are the same, the distance from each border point to the neighbor generators of p_1 can be replaced by the distance from the border point to p_1, which can be easily calculated based on the subgraph inside $V(p_1)$. As a result, the distances from q to each Voronoi neighbor of p_1 is equivalent to the sum of the distances from q to a border point and from that border point to p_1. For example, in Figure 3, the distance from q to p_3: $d_n(q, p_3) = d_n(q, b_3) + d_n(b_3, p_1)$. In the case where there are more than one border point between two adjacent generators, we pick the one with the smaller distance (shortest path). For example, both b_6 and b_7 are border points between p_1 and p_2. The distance from q to p_2: $d_n(q, p_2) = min(d_n(q, b_6) + d_n(b_6, p_1), d_n(q, b_7) + d_n(b_7, p_1))$. Note that the distances computed on the subgraph inside $V(p_1)$ are not necessarily the shortest distances from q to all neighbor generators on the original graph $\mathbb{G}$. However, for the generator which is the second NN of q, the shortest path distance computed on the subgraph inside $V(p_1)$ is identical to the shortest distance on the original graph $\mathbb{G}$. (Lemma 1 & Lemma 2).

Lemma 3 can be also generalized to the verification of distance from a query point to its k^{th} nearest neighbors on a road network.

Lemma 4. *Given a query point q, the $k-1$ nearest neighbors p_1 $, p_2, \ldots, p_{k-1}$, the union of subgraphs inside $V(p_1)$, $V(p_2), \ldots, V(p_{k-1})$, and the Voronoi neighbors of the $k-1$ NNs, the shortest path distance from q to the k^{th}NN can be verified.*

Proof: This is a generalization of Lemma 3. Given the network Voronoi cells of all $k-1$ NNs of p, by union all the road segments inside these Voronoi cells, we get a connected network g which is a subgraph of the whole road network $\mathbb{G}$. The shortest path from q to all the border points on the subgraph g can be computed using Dijkstra's algorithm. Next, as the distances from the border point to its two neighbor generators are the same, the shortest distance from q to all the neighbors of p_1 $, p_2, \ldots, p_{k-1}$ can be calculated based on g. Again, the shortest path distances computed based on the subgraph g are not necessarily the same as the shortest path distances on the original graph $\mathbb{G}$. However, for the k^{th}NN, p_k, Lemma 2 ensures that the shortest path distance between q and p_k on the subgraph g and the original graph $\mathbb{G}$ are the same.

Consequently, given a query point q, the k nearest neighbors p_1 $, p_2, \ldots, p_k$, the union of subgraphs inside $V(p_1)$, $V(p_2), \ldots, V(p_k)$, and the Voronoi neighbors of the k NNs, both shortest path (according to Lemma 1 & Lemma 2) and distance (according to Lemma 3 & Lemma 4) from q to the k^{th}NN can be verified.

5. CONCLUSIONS

In this paper, we studied the query integrity problem for k-nearest-neighbor queries on outsourced road networks and points of interest databases. While existing approaches proposed in this domain cannot verify both the distance and the shortest path to the kNN results simultaneously, we present a network Voronoi diagram based verification approach that utilizes the network Voronoi cell of each result object to verify the correctness and completeness of the kNN result with regards to both distance and path. We plan to extend our solution to handle other types of spatial queries on road networks in the future.

6. ACKNOWLEDGEMENTS

This research has been funded in part by the National Science Foundation grants CNS-0831505 (CyberTrust), CNS-0831502 (CyberTrust), CNS-0855251 (CRI), IIS-1115153, the USC Integrated Media Systems Center (IMSC), and unrestricted cash and equipment gifts from Northrop Grumman, Google, Microsoft and Qualcomm.

7. REFERENCES

[1] W. Cheng and K.-L. Tan. Query Assurance Verification for Outsourced Multi-dimensional Databases. *Journal of Computer Security*, 17(1):101–126, 2009.

[2] L. Hu, W.-S. Ku, S. Bakiras, and C. Shahabi. Verifying Spatial Queries Using Voronoi Neighbors. In *GIS*, pages 350–359, 2010.

[3] L. Hu, W.-S. Ku, S. Bakiras, and C. Shahabi. Spatial Query Integrity with Voronoi Neighbors. *IEEE Trans. Knowl. Data Eng.*, To appear, 2012.

[4] M. R. Kolahdouzan and C. Shahabi. Voronoi-Based K Nearest Neighbor Search for Spatial Network Databases. In *VLDB*, pages 840–851, 2004.

[5] W.-S. Ku, L. Hu, C. Shahabi, and H. Wang. Query Integrity Assurance of Location-Based Services Accessing Outsourced Spatial Databases. In *SSTD*, pages 80–97, 2009.

[6] W.-S. Ku, L. Hu, C. Shahabi, and H. Wang. A Query Integrity Assurance Scheme for Accessing Outsourced Spatial Databases. *GeoInformatica*, To appear, 2012.

[7] E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and Integrity in Outsourced Databases. *TOS*, 2(2):107–138, 2006.

[8] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* Probability and Statistics. Wiley, NYC, 2nd edition, 2000.

[9] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios. Authenticated Indexing for Outsourced Spatial Databases. *VLDB J.*, 18(3):631–648, 2009.

[10] M. L. Yiu, Y. Lin, and K. Mouratidis. Efficient Verification of Shortest Path Search via Authenticated Hints. In *ICDE*, pages 237–248, 2010.

[11] M. L. Yiu, E. Lo, and D. Yung. Authentication of Moving kNN Queries. In *ICDE*, pages 565–576, 2011.