The Area Code Tree for Nearest Neighbour Searching

Fatema Rahman and Wendy Osborn
Department of Mathematics and Computer Science
University of Lethbridge
Lethbridge, Alberta
T1K 3M4 Canada
Email: (f.rahman,wendy.osborn)@uleth.ca

***Abstract*—In this paper, we propose the Area Code Tree for nearest neighbour searching. It is a trie-type structure that stores points of interest (POIs) that are represented in area code format. An area code is a sequence of digits that indicate the relative location in space of a POI. We present the area code calculation, insert, and nearest neighbour search algorithms. We also present the results of a preliminary evaluation on the nearest neighbour search performance. We find that the efficient search times make the Area Code Tree an excellent candidate for continuous nearest neighbour processing for location-based services.**

***Keywords*—nearest neighbour search, spatial access methods, location-based services, performance**

I. Introduction

Location based services (LBSs) are services that return results based on certain location information [1] and query types. On such type of query is a nearest neighbour query.

Due to the rapid development of wireless technology and mobile devices, LBSs have become very popular. The environment of LBSs is classified into three types according to the mobility of the clients and the data objects. One of these types is for moving clients and static data objects, where the user continuously changes their location and submits a new query to the server for getting updated static data information. One such type of location-based application that assumes a moving query and static data are tourist information services [2], [3], [4], [5], [6]. continuous nearest-neighbour query. In a tourist information service, a tourist uses a location-aware mobile device to issue a continuous query for their closest Points of Interest (POIs). The POIs are managed on a separate server, so the query must be sent through a wireless network to have to server process the query and transmit the result back to the User. As the user moves around, their nearby POI changes, so this information must be updated continuously on their device.

One approach to provide up-to-date information is to pose a new query to the server after a position update. For mobile devices that have limited storage capacity, this may be the only option for performing a continuous spatial query. Therefore, it is important to have efficient query processing - in particular, for nearest neighbours - on the server. A spatial access method can be used to help speed up individual queries. Several approaches have been proposed that utilize spatial access methods [7], [8], [9], including [10], [11], [12], [13], [14], [15], [6]. Limitations of these approaches include caching a significant amount of data on the mobile device in order to avoid searching, and maintaining a sparse index which leads to inefficient searches.

978-1-4673-7788-1/15/$31.00

We propose a new data structure called an Area Code Tree for managing point data, such as POI data. The Area Code Tree is a trie-type structure in which all POIs are represented by an area code. An area code is a sequence of digits that represent the relative location of a point, such as a POI, in space. Therefore, indexing area code data, instead of actual spatial values, such as longitude and latitude, reduce the search for the nearest neighbour to a sequence of simple numeric comparisons. We present the algorithms for computing area codes, inserting a POI using its area code, and nearest neighbour searching. In addition, a preliminary performance evaluation for nearest neighbour searching is presented, which shows the search efficiency for different index sizes and data densities.

II. The Strategy

In this section we present the Area Code Tree and its associated area code calculation, insertion and nearest neighbour search algorithms. We first present some preliminaries before presenting the algorithms.

The Area Code Tree is a trie-type structure that organizes and manages points of interest (POIs) in area code form. In the algorithms below, a point of interest (POI) is represented as (Px,Py). A minimum bounding rectangle (MBR) defines a *region* of space. The **main region** defines the original space that is occupied by POIs. This region is reduced in size as the area code for a POI is calculated. It is represented with different coordinates: 1) the lower left coordinate (MINPx, MINPy), the upper right coordinate (MAXPx, MAXPy), and its centre (Cx, Cy). The centre divides a region into 4 regions and corresponding individual area codes: SW (1), SE (2), NW (3), NE (4). Therefore, the area code for a POI is a sequence of these individual area codes.

A. Area Code Calculation for a Point of Interest (POI)

Before calculating an area code for a POI, the lower left (MINPx,MINPy) and upper right (MAXPx,MAXPy) coordinates must be calculated from the set of POIs, in order to

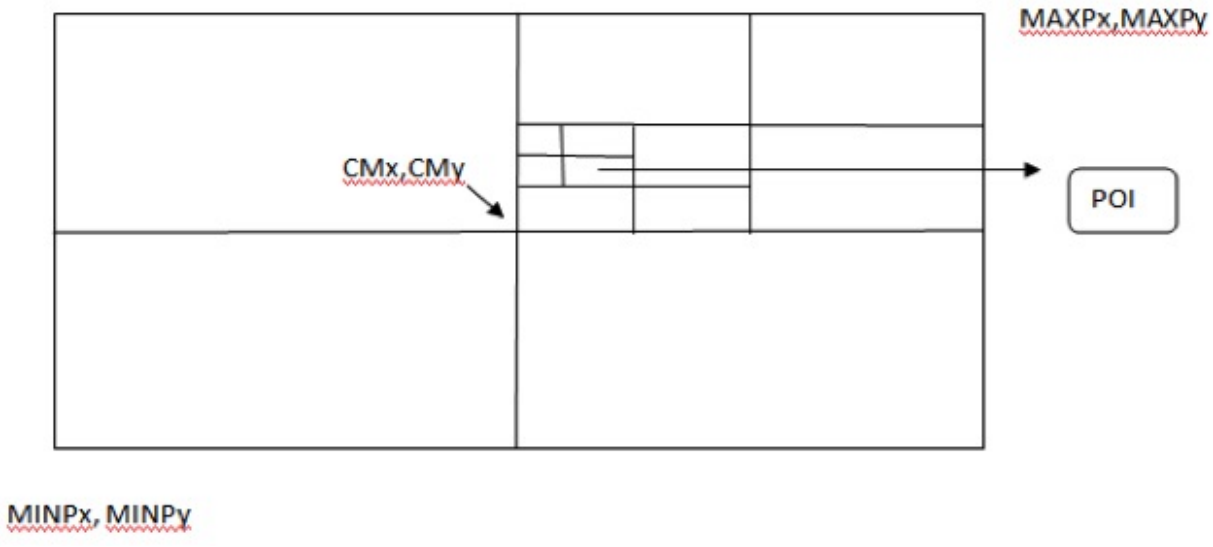


Fig. 1. Area Code Mapping for POI

define the main region. Then, the centre of the main region (CMx,CMy) is calculated.

Now, for each POI we calculate its area code by comparing it with the centre point (CMx,CMy) of the main region.

- *Case 1: Px < CMx and Py < CMy*. The POI is located SW of the centre in the main region, and therefore an initial area code of 1 is assigned to the POI. The minimum bounding rectangle of the main region will be updated as follows: The lower left coordinate will remain as (MINPx, MINPy), while the upper right coordinate of the region will be updated to (CMx,CMy), the centre point of the previous region (i.e. MAXPx= CMx, and MAXPy= CMy).
- *Case 2: Px > CMx and Py < CMy*. The POI is located SE of the centre in the main region, and an initial area code of 2 is assigned to the POI. The minimum bounding rectangle of the main region will be updated as follows: the new lower left corner of the region will be (CMx, MINPy) (i.e. MINPx = CMx and MINPy = MINPy), while the new upper right corner of the region will be (MAXPx, CMy) (i.e. MAXPx = MAXPx and MAXPy = CMy).
- *Case 3: Px < CMx and Py > CMy)*. The POI is located NW of the centre of the main region, and therefore an initial area code of 3 is assigned to the POI. The main region will be updated as: the new lower left coordinate will be (MINPx, CMy) MINPx= MINPx and MINPy = CMy. The new upper right coordinate of the main region will be (CMx, MAXPy) (i.e. MAXPx= CMx and MAXPy= MAXPy).
- *Case 4: Px > CMx and Py > CMy*. The POI is located NE the centre of the main region. The main region will be updated as: the lower left coordinate of the region will be (CMx, CMy) (i.e. MINPx= CMx and MINPy = CMy), while the upper right coordinate of the updated region will remain as (MAXPx, MAXPy).

Now, the new centre of updated region (CMx,CMy) will be calculated. The POI (Px, Py) will be compared with (Cmx,CMy). The next area code digit will be identified, and the lower left (MINPx,MINPy) and upper right (MAXPy,MAXPy) coordinates will be updated accordingly.

The above comparison and region update will continue until either: 1) the POI (Px,Py) is equal to the centre (CMx,CMy), or 1) the lower left (MINPx,MINPy) and upper right (MAXPy,MAXPy) coordinates are equal.

Figure 1 depicts an example area code mapping of for the POI shown in the diagram. Beginning with the main region (shown as the overall minimum bounding rectangle represented by (MINPx, MINPy) and (MAXPx,MAXPy) in the diagram), the POI is found to be located NE of (Cmx,CMy), and therefore is assigned an initial area code of 4. Continuing in the NE location, the POI is found to be located SE of the updated centre, and is assigned an additional area code of 1. For the remaining two updated regions, the POI is found to be NW and SE, respectively, of the updated centre. Therefore, the two remaining area codes will be 3 and 2 respectively. Therefore, the overall area code for the POI in the diagram is 4132.

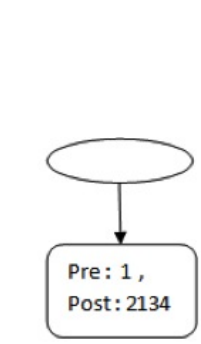
B. Area Code Insertion

Given the area code for a new POI, we insert the POI into into the Area Code Tree in the following manner. Beginning at the root node, we take the first digit from the area code and check if a path to a child node exists for that digit. If no such path exists, then a new child (leaf) node is created for that digit and the remainder of the area code is placed in the node. If a path to a child node exists for that digit, then the search for an insertion path continues to the corresponding child node, and the search for an insertion location continues with the next digit in the area code of the POI.

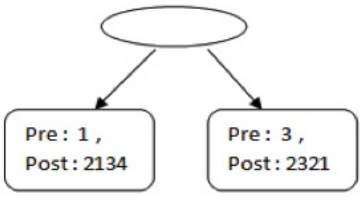
After a new leaf node is created, there may exist subsequent area code digits that two or more POIs share in common. In

TABLE I. SAMPLE AREA CODES

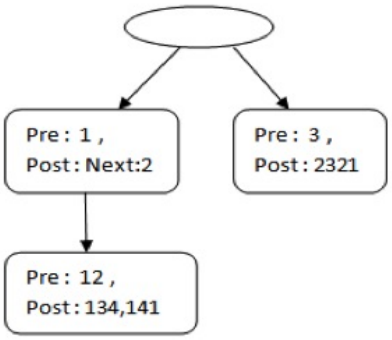
POI	Area Code
A	12134
B	32321
C	12141
D	32114
E	21324



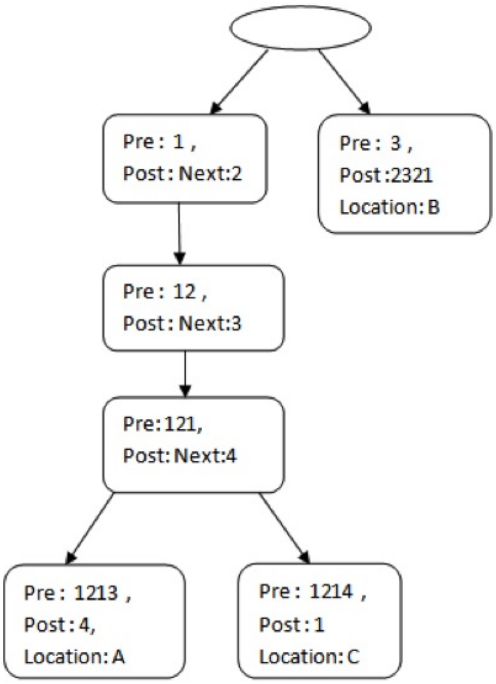
(a) After Insertion of POI A



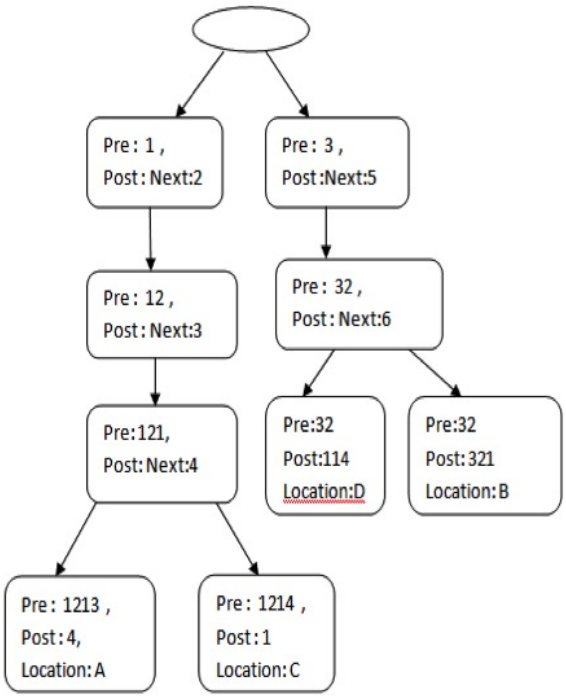
(b) After Insertion of POI B



(c) During Insertion of POI C



(d) After Insertion of POI C

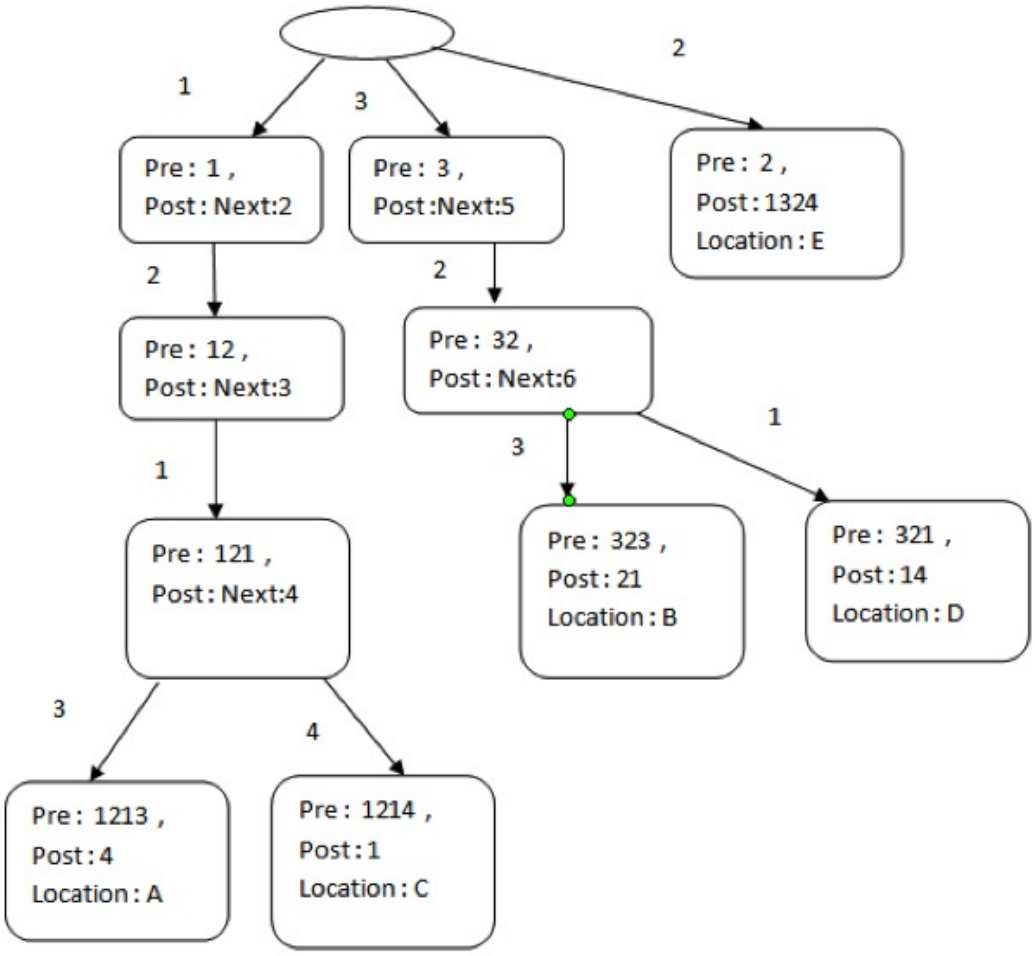


(e) After Insertion of POI D

Fig. 2. Index Construction via Insertion - Part 1

this case, further child nodes are created until each POI has its own leaf node.

We present some example insertions next. Table I gives some example area codes that are inserted into the area code tree shown in Figures 2 and 3. Beginning with POI A, it is inserted into a new child because it is the first POI to be inserted. Next, we insert POI B. The first digit of its area code is 3, which does not appear on an existing path in the Area Code Tree. Therefore, a new node is created to contain POI B. Figures 2a and 2b depicts the Area Code Tree thus far. There are two paths from the root - one for POI A (digit 1, Figure 2a) and the other for POI B (digit 3, Figure



(a) After Insertion of POI E

Fig. 3. Index Construction via Insertion - Part 2

reftree1).

Next we insert POI C. This is depicted across two figures - Figures 2c and 2d. The first area code digit for POI C, which is 1, has an existing path from the root. Therefore, the search proceeds in the child node. Since the child node is a leaf node, a new path and leaf node that corresponds to the second digit in the area code for POI C, which is 2, is created. This is depicted in Figure 2c. However, both POIs that are now referenced by the new leaf node - A and C - have the same third area code digit, which is 1. Therefore, another path and leaf node that corresponds to the common third digit is created. From here, two more paths and leaf nodes are created using the fourth area code digits for POIs A and C respectively. The resulting tree is depicted in Figure 2d.

Similarly, POIs D (Figure 2e) and E (Figure 3a) are inserted into the Area Code Tree. The final tree is depicted in Figure 3a.

C. Nearest Neighbour Searching

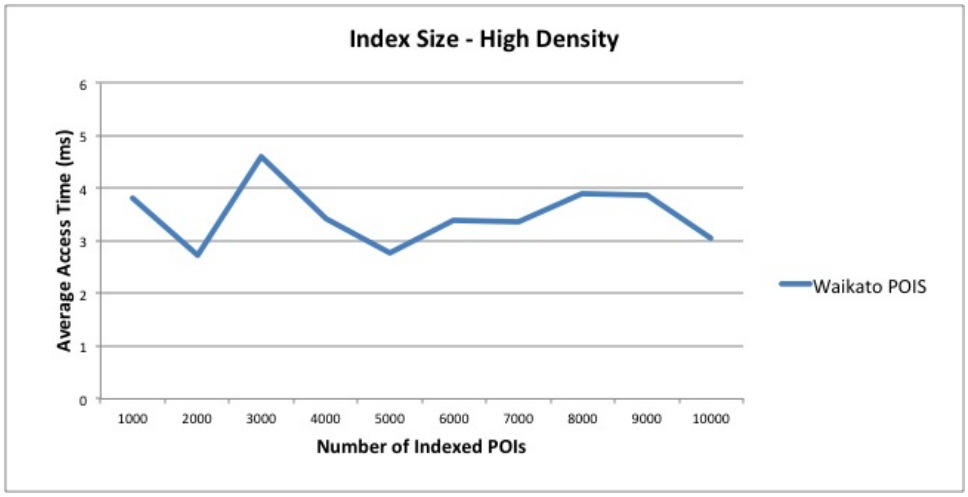
Because the POIs in the Area Code Tree are inserted using area code digits, the task of performing a nearest neighbour search is reduced to numeric comparisons. Given the area code for the current location of the User, the search begins at the root node. If a path from the root node exists that matches the first digit of the User's area code, then the search continues from the corresponding child node. If no such path exists, then the path with the closest path number is chosen. although this may be possible at any level of the Area Code Tree, it is very unlikely to occur in an index that is managing many POIs, and more likely to happen at the leaf levels.

The search continues in a similar manner until a leaf node is reached, and the POI stored at that leaf node is returned for the result. Given a User's area code of 12135 and the Area Code Tree in Figure 2e, the search proceeds along the path identified by 1, then 2, then 1, then 3, and finally reaching the node for POI A.

III. Evaluation

In this section, we present the framework and results of our empirical evaluation. We evaluate the search performance of the area code tree. We first present the datasets we utilized and tests we performed to evaluate our structure. Then, we present and discuss the results of our evaluation.

(a) POIs over North Island



(b) POIs over Waikato

Fig. 4. Evaluation Results

A. Data Sets

For our experiments, we generated twenty-one data sets overall that represent different points across New Zealand. Ten of those data sets consist of random POIs drawn from across the North Island of New Zealand. Each file contains 1000, 2000, 3000,..., up to 10,000 POIs respectively. Another ten data sets consist of random POIs drawn from across the Waikato Region of New Zealand. Also in this case, each file contains 1000, 2000, 3000,..., up to 10,000 POIs respectively. The Waikato Region is a smaller area that is part of the North Island. Therefore, these data sets are denser and will allow us to evaluate how quickly the Area Code Tree can be used to find a nearest neighbour in denser data.

Finally, the remaining data set consists of 10 User locations along their trajectory. They will serve as a series of continuous nearest neighbour queries for our evaluations.

B. Experiments

We performed 200 nearest neighbour evaluations in the following manner. First, an Area Code Tree was created for each of the 20 data sets mentioned above. Next, for each tree, ten nearest neighbour searches were performed using the User location point set. The performance criteria that was measured for each of the 200 searches was the execution time. Finally for each tree, the average search time was calculated.

C. Results

Figure 4 depict the results of our search evaluation. We find that the amount of time required for performing a nearest neighbour search in the Area Code Tree is between 2ms and 5ms on average. We have made the following observations. First, it appears that the number of points in the index does not significantly affect the time it takes to locate the nearest neighbour for a User location. In addition, we achieve similar results in both the denser and not-so-dense set of points, so

density does not appear to affect the outcome of the nearest neighbour search.

IV. Conclusion

We present the Area Code Tree for nearest neighbour search, which stores points of interest (POIs) that are represented in area code format. We present the area code calculation, insert, and nearest neighbour search algorithms. We also present the results of a preliminary evaluation on the nearest neighbour search performance. We find that the efficient search times - between 2ms and 5ms on average, and regardless of the density of the dataset and the number of POI area codes in the index. This makes the Area Code Tree an excellent candidate for continuous nearest neighbour processing for location-based services. Some future research directions include the following. The first is to further evaluate the search performance by comparing the Area Code Tree to other spatial access methods for nearest neighbour and continuous nearest neighbour search. The second is to evaluate the performance of the insertion algorithm for constructing the Area Code Tree. In addition, a further research direction is to explore the ability to perform k nearest neighbour searches using the Area Code Tree. Finally, it would be interesting to explore the use of the Area Code Tree for other types of area codes, such as postal codes of various countries.

Acknowledgment

The authors would like to thank the referees for their constructive recommendations for improvement and future work.

References

[1] J. H. Schiller and A. Voisard, Eds., *Location-Based Services*. Morgan Kaufmann, 2004.

[2] S. Poslad, H. Laamanen, R. Malaka, A. Nick, P. Buckle, and A. Zipf, "Crumpet: Creation of user-friendly mobile services personalized for tourism," in *Proc. IEE 3G2001 Mobile Communication Technologies Conf.*, London, UK, 2001.

[3] P. Klante, J. Krshe, and S. Boll, "Accessignts - a multimodal location-aware mobile information system," in *Proc. 9th Int'l Conf. on Computers Helping People with Special Needs*, Paris, France, July 2004.

[4] A. Hinze, A. Voisard, and G. Buchanan, "TIP: Personalizing information delivery in a tourist information system," *Journal of IT & Tourism*, vol. 11, no. 3, pp. 247–264, 2009.

[5] A. Hinze and Q. Quan, "Trust- and location-based recommendations for tourism," in *Cooperating Information Sytsems (Coopis)*, 2009, pp. 414–422.

[6] W. Osborn and A. Hinze, "Tip-tree: A spatial index for traversing location in context-aware mobile systems to digital libraries," *Pervasive and Mobile Computing*, 2013, in press.

[7] V. Gaede and O. Günther, "Multidimensional access methods," *ACM Computing Surveys*, vol. 30, pp. 170–231, 1998.

[8] S. Shekhar and S. Chawla, *Spatial Databases: A Tour*. Prentice Hall, 2003.

[9] P. Rigaux, M. Scholl, and A. Voisard, *Spatial Databases: with Application to GIS*. Morgan-Kauffman, 2001.

[10] Z. Song and N. Roussopoulos, "K-nearest neighbor search for moving query point," in *Proc. 7th Int'l Symp. on Advances in Spatial and Temporal Databases*, 2001, pp. 79–96.

[11] Y. Tao, D. Papadias, and Q. Shen, "Continuous nearest neighbor search," in *Proc. 28th Int'l Conf. Very Large Data Bases*, 2002, pp. 287–298.

[12] H. Hu, J. Xu, W. Wong, B. Zheng, D. Lee, and W.-C. Lee, "Proactive caching for spatial queries in mobile environments," in *Proc. 21st Int't Conf. on Data Engineering*, 2005.

[13] H. Jung, S.-W. Kang, M. Song, S. Im, J. Kim, and C.-S. Hwang, "Towards real-time processing of monitoring continuous k-nearest neighbour queries," in *Proc. 2006 Int'l Conf. Frontiers of High Performance Computing and Networking*, 2006.

[14] K. Lee, W.-C. Lee, H. Leong, B. Unger, and B. Zhang, "Efficient valid scope computation for location-dependent spatial queries in mobile and wireless environments," in *Proc. 3rd Int'l. Conf. Ubiquitous Information Management and Communication*, 2009.

[15] Y. Park, K. Bok, and J. Yoo, "An efficient path nearest neighbour query processing scheme for location-based services," in *Proc. 17th Int'l Conf. Database Systems for Advanced Applications*, 2012.