The Area Code Tree for Approximate Nearest Neighbour Search in Dense Point Sets

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

Abstract

In this paper, we present an evaluation of nearest neighbour searching using the Area Code tree. The Area Code tree is a trie-type structure that organizes area code representations of each point of interest (POI) in a data set. This data structure provides a fast method for locating an actual or approximate nearest neighbour POI for a query point. We first summarize the area code generation, insertion (used in overall construction) and searching approaches. Then, we evaluate the contraction, insertion, searching, and accuracy of the Area Code tree, with some of the evaluation involving the comparison versus a basic benchmark brute force approach. We find that when the Area Code tree is used for locating an approximate nearest neighbour, that low constant-time search is achieved, and in denser POI sets, higher accuracy is achieved. This ultimately makes the Area Code tree a strong candidate for approximate continuous nearest neighbour processing for location-based services.

keywords: nearest neighbour queries, spatial access methods, location-based services

1 Introduction

A location-based service provides results to a user of a mobile device (e.g. smartphone, tablet) based on their location, interests and the type of query being performed [10]. One example of such a query is a nearest neighbour query [9, 11], which returns the nearest point of interest (POI) to them. For example, a user may want to know the location of the nearest restaurant to them. The user may want to know the exactly closest restaurant to them. However, the user may also be happy with another suggestion that, although not guaranteed to be the closest, may be close enough to satisfy them. This is an example of an approximate nearest neighbour, where a trade-off is being made between accuracy and efficiency.

Efficient nearest neighbour processing - exact or approximate - is important, but is especially important when it is initiated from a mobile device [10]. Many strategies have been proposed for nearest neighbour processing for location-based services. Several utilize spatial access methods [1], including [12, 13, 3, 4, 5, 7, 6]. Although all of these strategies return exact nearest neighbour, limitations of these approaches include repeated searching, the need to cache a significant amount of data on the mobile device, the requirement to know the query trajectory in advance, and maintaining a sparse index which leads to inefficient searches.

Repeated searching, although not desirable, may be the only option available when storage on a mobile device is limited. A recently proposed data structure, the Area Code tree [8] stores and manages POIs in a trie-type structure using an area code representation for each POI. Although it can be used to locate POIs efficiently, it cannot be used for exact nearest neighbour matching. However, given the preliminary evaluation on its efficiently, it is an excellent candidate for approximate nearest neighbour search for situations where a guaranteed exact answer is not required - for example, in the restaurant finding scenario given above.

Therefore, in this paper, we evaluate the Area Code tree for accuracy, tree construction time, and also comparatively evaluate the search time against another strategy. We find that approximate nearest neighbour searching can be accomplished in very low and constant time, regardless of the number of POIs being indexed. With respect to accuracy, up to 60% accuracy is achieved when the Area Code tree is used for indexing dense POI sets. This make the Area Code tree a significant candidate for continuous approximate nearest neighbour search for location-based services.

The remainder of the paper proceeds as follows. Section 2 summarizes related work in the area of continuous nearest neighbour processing for mobile

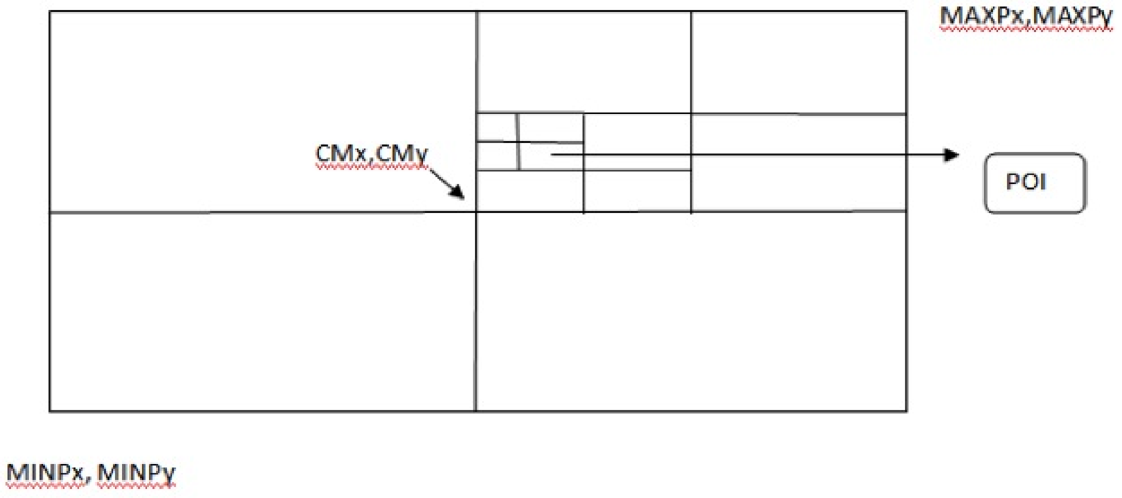

Figure 1: Area Code Mapping (from [8])

devices. Section 3 summarizes the area code mapping, insertion and search algorithm for the Area Code tree. Section 4 presents the methodology and results of our performance evaluation. Finally, Section 5 concludes the paper and provides some directions of future research.

2 Related Work

In this section, we summarize related work in nearest neighbour searching for location-based services. Although nearest neighbour strategies have been proposed in other contexts, they are considered outside of the scope of this work. Many strategies have been proposed in the literature [12, 13, 3, 4, 5, 7, 6].

A continuous nearest neighbour strategy proposed by Song and Roussopoulos [12] obtains a superset m of nearest neighbours, which attempts to keep the result current while the query point moves around, and a new query call to the server is not necessary. Their strategy utilizes existing stationery nearest neighbour strategies. A limitation of their strategy is in choosing the value of m so that fewer query calls are needed but not too much data needs to be stored on the mobile device.

Tao *et al.* [13] utilize the R-tree [2] to speed up the repeated searching needed for their continuous query strategy. Lee *et al.* [5] improve upon this strategy by fetching both the required and some additional objects, in order to reduce the number of repeated searches that are needed. Park *et al.* [7]also improve upon this strategy by locating all nearest neighbours along a trajectory by using the R-tree. A limitation exists in that the trajectory needs to be known in advance.

Hu *et al.* [3] propose a proactive caching strategy, which caches previous results and the R-tree nodes required to obtain them on the mobile device. The cache is always searched first for a new query, with additional results fetched from the server. Limitations include the significant overhead of caching and local processing on the mobile device.

Jung *et al.* [4] utilize a grid index for continuous nearest neighbour searching. The grid index allows for quick elimination of regions of space from consideration if they do not overlap the query point. A limitation with this strategy is that the grid index can be sparse due to wasted space.

In summary, some general limitations of these approaches include caching a significant amount of data on the mobile device, repeating searching, using a sparse index which leads to inefficient searches, and requiring knowledge of the query path in advance. Repeated searching, however, may be the only option if storage is limited on a mobile device. The Area Code tree [8] attempts to provide the ability to perform repeated searching that is efficient, but at the cost of accuracy. It is also more compact that existing spatial access methods, given that only POIs are stored, and not many co-ordinates for many bounding rectangles. The Area Code tree is summarized in the next section.

3 Area Code Tree

The Area Code Tree [8] is a trie-type structure for approximate nearest neighbour searching. It stores points of interest (POIs) that are represented in an area code format. In this section, we briefly summarize

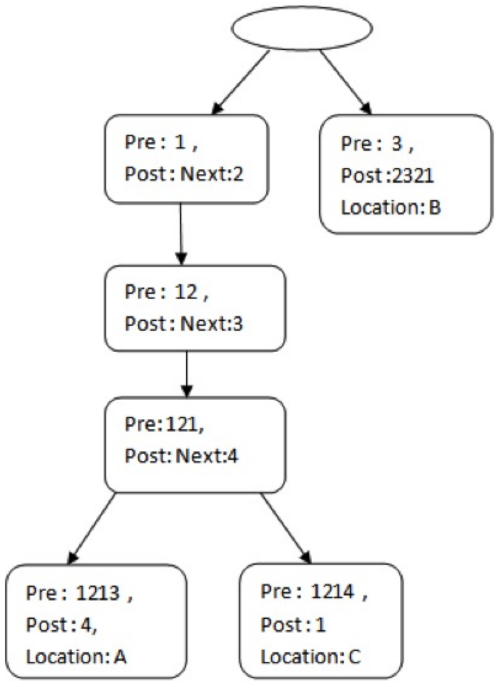

Figure 2: Area Code Tree after inserting A,B,C (from [8])

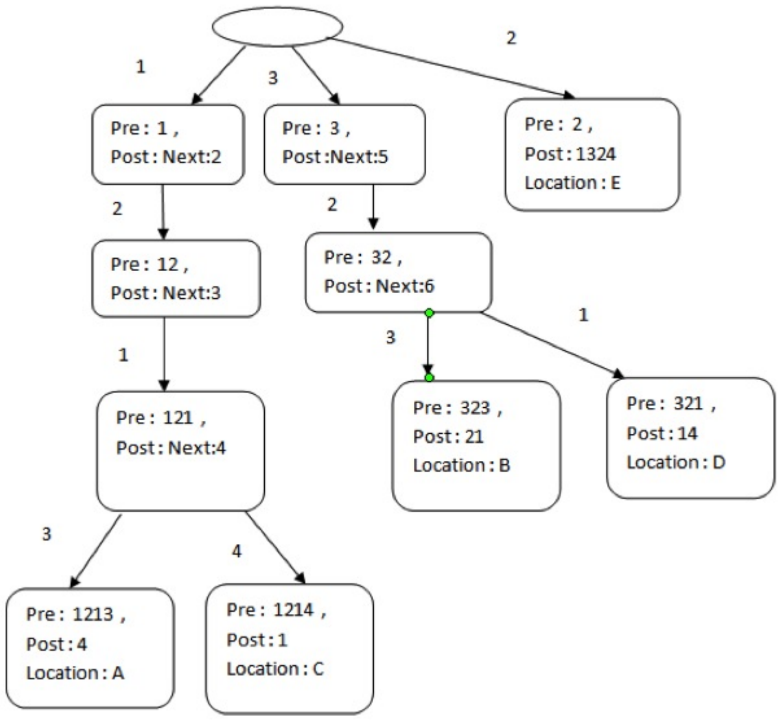

Figure 3: Area Code Tree after inserting D,E (from [8])

the mapping, construction and nearest neighbour search for the Area Code tree.

An area code is a sequence of digits that indicate the relative location of a POI in space. It is obtained by recursively partitioning the space containing points into quadrants. For a particular POI, the space is partitioned until the POI is equal to the middle of a quadrant. At each level of partitioning, the quadrants are numbered as follows: SW (1) SE (2) NW (3) and NE (4). Beginning with the top-most partition, a POI obtains a digit at each level, depending on which quadrant it resides in. Figure 1 depicts an example of space partitioning and mapping. The POI maps to the area code 4132, since the POI resides in the top-most NE quadrant, followed at the next level by the SW quadrant, then the NW quadrant, and finally the SE quadrant.

Once the area code for a POI is determined, is it inserted into the Area Code Tree, beginning with the most significant area code digit. To construct a tree, each area code is inserted one at a time, using the existing strategy for any trie structure. Figures 2 and 3 depict a small Area Code Tree, containing the POIS A,B,C,D and E, and each with area codes 12134, 32321, 12141, 32114, and 21324 respectively.

Given that every POI is mapped to a string of digits, this provides a method for quickly identifying an approximate nearest neighbour to the query point. A search begins by first mapping the query point to an area code using the same strategy mentioned above. Then, beginning with the most significant digit of the query point area code, a path is followed down the Area Code tree while digits in the query match digits in the tree nodes. If a match does not exist at a particular level, the closest match is taken. For example, suppose we have the query area code of 12144. Referring back to Figure 3, the closest match is the POI with area code 12141, since a path exists that matches the first 4 digits of the query area code, with the closest node to the last digit in the query area code containing the value of 1.

4 Evaluation

In this section, we present the methodology and result of our performance evaluation of the Area Code tree. We compare its search performance with the Brute Force method, which consists of searching the set of POIs to find the nearest neighbour. We chose this comparison for our preliminary evaluation due to the Brute Force method being a basic benchmark for processing nearest neighbour queries. In addition, we evaluate the construction time and the accuracy of our proposed structure. Construction is performed by inserting one POI area code at a time, while accuracy is measured by determining the percentage of times that an accurate nearest neighbour is found when the Area Code tree is used.

For our evaluation, we use twenty-one synthetically generated data sets that represent collections of different POIs across New Zealand. Ten data sets consist of POIs drawn from the North Island of New Zealand, with each set containing 1000, 2000, 3000,..., up to 10,000 POIs respectively. An additional ten data sets contain POIs from the Waikato Region of New Zealand (part of the North Island), again with each file containing 1000, 2000, 3000,..., up to 10,000 POIs respectively. The Waikato data sets are denser, which allows us to evaluate the Area Code Tree in denser data. The remaining data set contains 10 User locations along their trajectory, which will serve as the nearest neighbour queries for our evaluation.

First, an Area Code Tree is created for each of the POI sets above. Then, for each tree, ten nearest neighbour searches are performed using the User location set. The same searches are also performed on the same data sets using the Brute Force method. Therefore, 200 nearest neighbour comparisons are performed.

The performance criteria that are measured are as follows:

- For each tree construction, both the overall construction time and the average insertion time per POI area code,
- For the search comparison, the average search time (in milliseconds).
- Accuracy of the Area Code tree, which is measured by recording the percentage of POIs found by the Area Code Tree that matched those found by the Brute Force search.

First, figures 4 and 5 contain the results of our comparison using the Waikato and North Island data sets, respectively. For both figures, the x-axis contains values that represent 1000s of POIs (i.e. 1 is 1000 POIs, up to 10 for 10,000 POIs), while the y-axis represents the average search time in seconds. We find for both groups of POI sets that the Area Code Tree results in significantly better search times over the Brute Force method. The average search time for the Area Code tree is less than 10ms, and is regardless of the density of the dataset and the number of POI area codes in the index. For the brute force method, the average search time increases linearly, between less than 10ms to up to almost 50ms for searching in 10,000 area codes.

The North Island and Waikato POI sets differ in the accuracy they achieve. Figure 6 depicts the results of

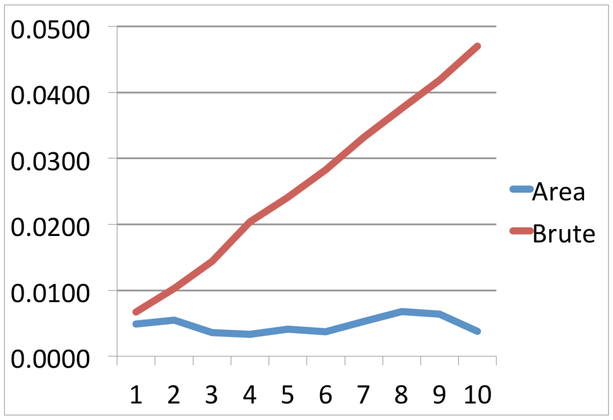

Figure 4: Waikato POIs

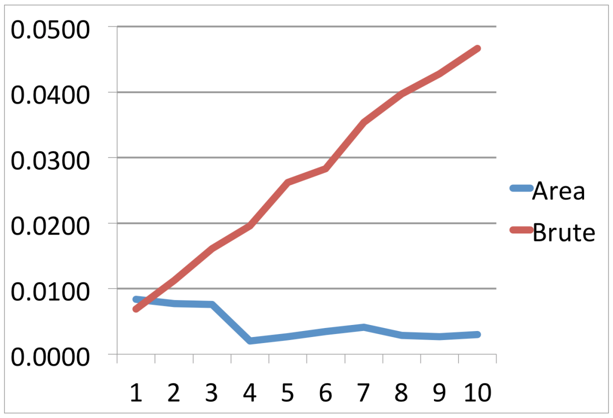

Figure 5: North Island POIs

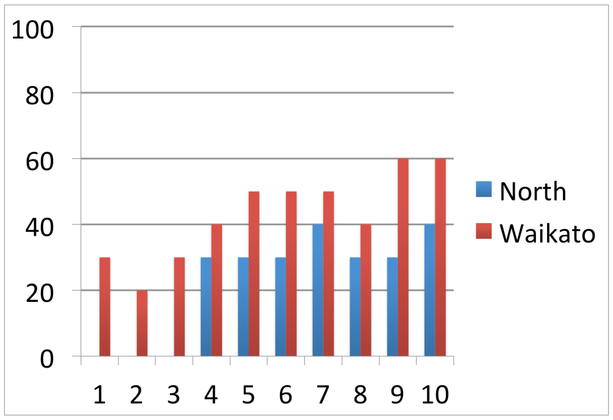

Figure 6: Accuracy of Area Code Tree

Data Sets	#POIs	O/A Time	Avg. Time
North Island	1000	46.16	0.046
	2000	79.60	0.040
	3000	106.61	0.036
	4000	128.21	0.032
	5000	166.32	0.033
	6000	219.94	0.037
	7000	281.70	0.040
	8000	334.07	0.042
	9000	437.50	0.049
	10000	490.96	0.049
Waikato	1000	18.63	0.019
	2000	39.71	0.020
	3000	91.74	0.030
	4000	159.69	0.040
	5000	227.16	0.045
	6000	303.42	0.051
	7000	409.23	0.058
	8000	481.41	0.060
	9000	586.86	0.065
	10000	686.63	0.069

Table 1: Tree Construction Times (in seconds)

the evaluation for accuracy. We find that for dense point sets (i.e Waikato), the Area Code Tree can achieve between 40% and 60% accuracy in locating the nearest neighbour. For the less dense point sets (i.e. North Island) however, the accuracy was lower - between 0% and 40%. Therefore, we conclude that, for denser point sets, the Area Code Tree provides fast nearest neighbour searching, that is expected to improve in accuracy as the data set size increases in density.

Finally, figure 1 depicts the construction times of the Area Code Tree for the various sets of POIs. Both the overall tree construction time (O/A Time) and the average time per insertion of a POI (Avg. Ins) are recorded. We do observe that the time it takes to construct an Area code tree via repeated insertion increases significantly with the size of the POIs. For the North Island datasets, the time ranges from just under a minute for 1000 POIs, up to just over 8 minutes for 10,000 POIs. For the Waikato Region datasets, the times range from well under a minute to over 10 minutes. Although these overall construction times seem high, two things must be noted. First, for static data sets, the Area Code tree only needs to be constructed once in order to be searched many times. Given the search performance above, this is a small price to pay. Second, if the occasional insertion needs to take place, on average this can take place in under 50ms for the less denser data sets, and under 70ms for the denser data sets, because a complete re-build of the tree is not required.

5 Conclusion

In this paper, we present an evaluation of the Area Code tree for accuracy, tree construction time, and also comparatively evaluate the search time against a Brute Force strategy. We find that approximate nearest neighbour searching can be accomplished in very low and constant time, regardless of the number of POIs being indexed. With respect to accuracy, up to 60% accuracy is achieved when the Area Code tree is used for indexing dense POI sets. This make the Area Code tree a significant candidate for continuous approximate nearest neighbour search for location-based services. The only costly factor is in the tree construction time. However, for fairly static data sets, this is a small price to pay for the savings in search time and increased accuracy (in some cases) that are achieved. Some future research directions include the following: 1) to further evaluate the search performance by comparing the Area Code Tree with other spatial access methods for continuous nearest neighbour search, 2) to extend the Area Code Tree to find k nearest neighbours, and 3) to utilize other types of "area codes", such as telephone area codes, and postal/zip codes, in the Area Code tree.

References

[1] V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30:170–231, 1998.

[2] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Int'l Conf. Management of Data*, pages 47–57, 1984.

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

[4] HR Jung, S.-W. Kang, MB Song, SJ 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.

[5] K.C.K. Lee, W.-C. Lee, H.V. 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.

[6] W. Osborn and Hinze. A. Tip-tree: a spatial index for traversing locations in context-aware mobile access to digital libraries. *Pervasive and Mobile Computing*, 15:26–47, 2014.

[7] 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.

[8] F. Rahman and W. Osborn. The area code tree for nearest neighbour searching. In *Proc. 2009 IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing*, 2015.

[9] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. *SIGMOD Rec.*, 24(2):71–79, May 1995.

[10] Jochen H. Schiller and Agnès Voisard, editors. *Location-Based Services.* Morgan Kaufmann, 2004.

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

[12] 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*, pages 79–96, 2001.

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