Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 155 (2019) 282–288

Procedia Computer Science

www.elsevier.com/locate/procedia

The 16th International Conference on Mobile Systems and Pervasive Computing (MobiSPC)
August 19-21, 2019, Halifax, Canada

Continuous Region Query Processing in Clustered Point Sets

Wendy Osborn[a,*], Farzin Keykavoos[a]

[a]*Department of Mathematics and Computer Science, University of Lethbridge, Lethbridge, Alberta, T1K 3M4, Canada*

Abstract

There exists an increasing usage rate of location-based information from mobile devices, which requires new query processing strategies. One such strategy is a continuous region query in which a moving user continuously sends queries to a central server to obtain data or information. In this paper, we introduce two strategies to process a spatial moving query over clustered data sets. Both strategies use a safe region approach on the client in order to minimize the number of queries that are sent to the server. We explore the use of a two-dimensional indexing strategy, as well as the use of Expectation Maximization (EM) and k-means clustering. Our experiments show that both strategies outperform a Baseline strategy where all queries are sent to the server, with respect to data transmission, running time and workload costs.

Keywords: location-based services; continuous region query processing; spatial access methods; k-means; expectation-maximization

1. Introduction

A Location Based Service (LBS) [12] provides up-to-date information on points of interest (POIs) to the user of a mobile device, based on their location. For example, a user is interested in tourist attractions (e.g., museums), that are located within a certain region from their current location. This information can be obtained by using a region query, which is sent from the user's mobile device to a central server to be processed, with the result then sent back to the user. This query could be answered quickly if the user remains stationary. However, issues arise with providing up-to-date information if the user is moving. If the user's location is changing, the location of the query region is also changing, which can produce a different result. This is a continuous region query, where the result must be updated as the user changes location. Keeping the result of a moving region query up-to-date is a challenging problem for LBSs. One strategy used in research for processing a continuous region query is a safe region. A safe region is a super-region (and the data within that region) that is chosen by the server to encompass the query region and reside on the user's mobile device. The idea behind it is that while a region query remains inside of it, the result remains valid

* Corresponding author. Tel.: +1-403-329-2294 ; fax: +1-403-317-2882.
E-mail address: wendy.osborn@uleth.ca

1877-0509
10.1016/j.procs.2019.08.040

and can be processed locally without having to send requests to the server. Several previously proposed strategies for processing a continuous region query use some type of safe region. One limitation among these strategies is not use of clustering. The use of a clustered dataset can help reduce the cost of generating a new safe region, while supporting an incremental strategy to user trajectory processing. Therefore, we propose two strategies for continuous region query processing that take advantage of the clustering of a dataset. Given a clustered dataset, our strategies create a clusterMBR (i.e. cluster minimum bounding rectangle (MBR)) representation for each. Using the clusterMBRs, a safe region (i.e superMBR) is created by locating qualifying clusterMBRs instead of locating qualifying points in a large point set. We explore in the second strategy the use of a spatial index with a more efficient search strategy, in order to improve performance. Our strategies are empirically evaluated against a baseline strategy to show the benefits of using a clustered point set.

2. Related Work

In this section, we provide a summary of existing strategies in the area of continuous query processing, with a focus on continuous region queries in static data sets. Tao et al. [14] use an R-tree for improving search performance. A limitation of this strategy is the repeated "vertical" searching for new co-ordinates. Park et al. [11] propose a "horizontal" search for nearest neighbour objects along a trajectory. A limitation is the user trajectory must be known in advance. Zhang et al. [16] use Retrieve-Influence-Sets (RIS) for forming safe regions for processing both continuous region and nearest neighbour queries. This work also requires the trajectory to be known in advance. Park et al. [10] propose a strategy for broadcast networks where broadcasted results are sorted by location. A limitation with this strategy is that clients must continuously tune-in to the channel in order to obtain results. Wu et al. [15] propose a memory-based strategy that uses a tile-based query index. A limitation with this strategy is identifying the appropriate tile size. Hu et al. [5] use caching to support multiple spatial query types. Their strategy caches on the mobile device results from previous queries and the R-tree nodes that lead to them. The cached partial R-tree is always searched first, before fetching additional objects from the server. Limitations include costly caching overhead and local processing. Jung et al. [6] uses a grid index, where every grid cell contains a minimum bounding rectangle (MBR) that contains the points in the cell. If a query does not overlap the MBR, none of its points will overlap. One limitation is that many cells will be sparse. Lee et al. [7] propose a strategy for reducing the number of required queries by fetching both required and additional complementary objects, using an R-tree. One issue is that repeated searching is still required to obtain enough complementary objects. Two strategies are provided by Al-Khalidi et al. [2, 1]. In [2], a Monte Carlo techinque is proposed for determining irregularly shaped safe regions. In [1] address the issue of identifying when a user will leave a safe region before it happens. Osborn [9] proposed a strategy that used in-structure searching of a spatial index, in order to significantly reduce or eliminate the need to perform repeated searching from the root. The mqrtree [8] is used due to its two-dimensional structure, which allows for better support of in-structure searching. A limitation with this approach is the need to traverse the structure from the current node in order to find the corresponding point set to send to the client. In summary, limitations of the above include: 1) requiring that the user's trajectory be known in advance, or requiring a costly re-build of the safe region, and 2) using a spatial index that uses inefficient search strategies, including traversals. The strategies proposed in the next section use a clustered dataset to help reduce the cost of generating a new safe region, while supporting an incremental strategy to continuous region query processing.

3. Strategies

This section presents our strategies for continuous region query processing. Both use a safe region, referred to as a super minimum bounding rectangle (superMBR) in our work, to reduce the number of queries that must be sent back to the server. We first present some preliminaries namely, the formation of cluster minimum bounding rectangles (clusterMBRs) and superMBRs. Following this, we then present our two strategies.

3.1. ClusterMBRs and the SuperMBR

A major building block for the superMBR is the clusterMBR. A clusterMBR is the minimum rectangular extent needed to encompass a set of points that form a cluster. After a set of clusters are formed for a point set, each cluster

118.32 :	4.83 :	141.42 :	77.61 :	cluster0
55.62 :	218.54 :	86.18 :	228.24 :	cluster1
1.75 :	38.69 :	56.01 :	85.18 :	cluster2
166.11 :	115.14 :	204.77 :	166.11 :	cluster3
212.15 :	55.62 :	228.24 :	100.89 :	cluster4
60.77 :	85.50 :	112.75 :	127.78 :	cluster4
100.89 :	177.06 :	154.39 :	212.15 :	cluster6
4.83 :	131.63 :	51.91 :	141.42 :	cluster7

Fig. 1. Cluster MBRs

118.32 :	77.61 :	118.32 :	77.61 :	1
123.34 :	69.35 :	123.34 :	69.35 :	2
127.79 :	60.78 :	127.79 :	60.78 :	3
131.64 :	51.95 :	131.64 :	51.95 :	4
134.87 :	42.81 :	134.87 :	42.81 :	5
137.48 :	33.51 :	137.48 :	33.51 :	6
139.45 :	24.05 :	139.45 :	24.05 :	7
140.76 :	14.48 :	140.76 :	14.48 :	8
141.42 :	4.83 :	141.42 :	4.83 :	9

Fig. 2. Point Set of Cluster 0

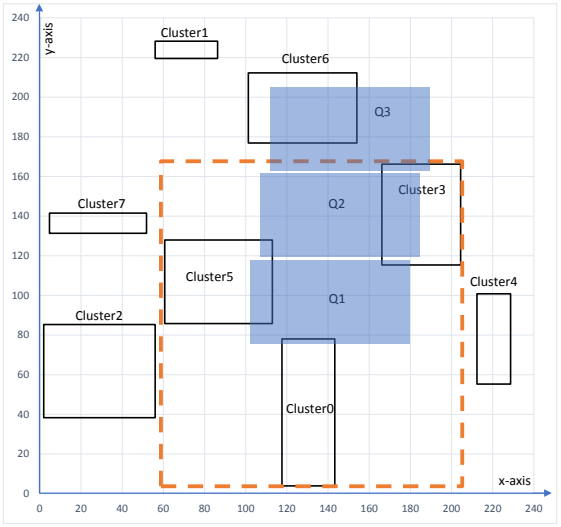

Fig. 3. First, Second, and Third Query

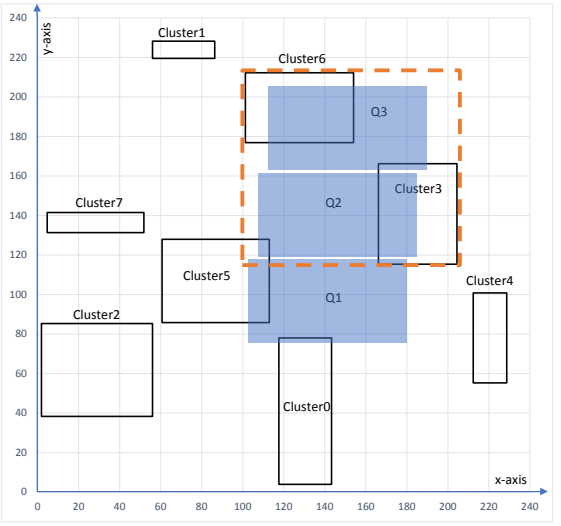

Fig. 4. New SuperMBR Resulting from Third Query

will have a cluster MBR formed for it, which will serve as a representation of the cluster. Figs. 1 and 2 depict an example set of clusterMBRs for a set of points, and an example set of points for one of those clusterMBRs (namely, Cluster 0). In both figures, the first two values (lx, ly) are the lower-left corner of an MBR, while the next two values (hx, hy) are the upper-right corner of the MBR. In the case of Fig. 2, both x values and both y values are the same, since a point is being represented. Finally, an identifier is provided for the point or cluster MBR. The superMBR will serve as the safe region for our strategy. It is the minimum rectangular extent needed to encompass some superset set of points surrounding a region query. Our approach to creating a superMBR is the following. First, the set of clusterMBRs is searched to identify those that overlap the region query. Then, the superMBR is created by finding the minimum rectangular extent that includes the clusterMBRs. Finally, the clusters corresponding to each chosen clusterMBR are fetched before the superMBR and all fetched points are sent to the client. Using the clusterMBRs as the basis for the superMBR, instead of the point set directly, helps to speed up the process of creating a safe region, as the set of clusterMBRs is significantly smaller than the entire set of point. In addition, no decisions on the size of the superMBR needs to be made, since this ultimately is dictated by the size and location of the clusterMBRs. Fig. 3 shows superMBRs for queries Q1, Q2 and Q3. We start with Q1. Here, we can see that Q1 overlaps the clusterMBRs for clusters 0, 3 and 5. The resulting superMBR encompasses these three clusterMBRs. Therefore, as long as Q1 remains within the bounds of the superMBR, a new query request to the server will not be required.

3.2. Strategies

We now present the two strategies for continuous region query processing. In the First strategy, the client device sends the first query region request to the server. The server creates a superMBR and corresponding point set to send

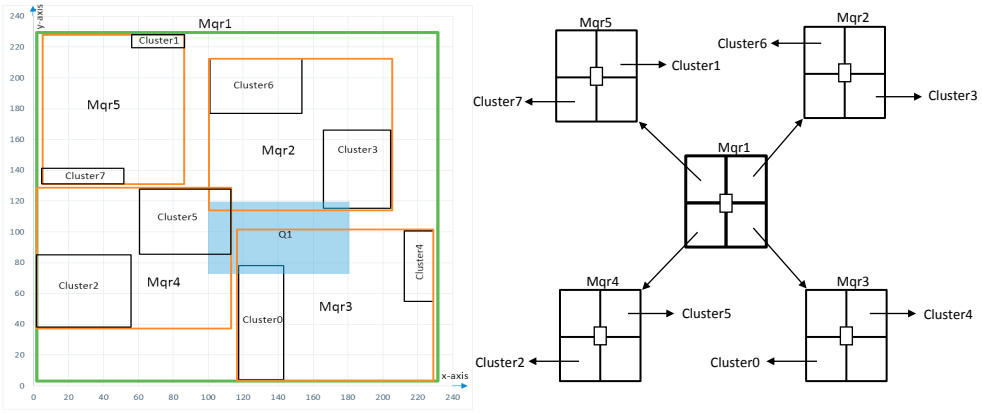

Fig. 5. mqr-tree containing cluster MBRs

back to the client so that query processing can continue on the local device while the query remains in the superMBR. This is carried out using the following strategy: 1) a **linear search** is performed on the set of clusterMBRs to find those that overlap the query region, 2) a superMBR is formed that encompasses the chosen clusterMBRs, 3) the points for each clusterMBR are fetched, and 4) the superMBR and point set are sent to the client.

On the client, the points that reside inside the query region are determined and displayed and sent to the client. For each subsequent update to the query region on the client, the query region will be processed by one of the following options. First, if the updated query region still resides fully inside of the current superMBR, then the result can be produced on the client, and no request needs to be sent to the server. Otherwise, the subset of points on the client is no longer guaranteed to be complete for the query, so a new region query request needs to be sent to the server in order to receive an updated superMBR and corresponding point set. To illustrate, we present a short example that shows the above strategy. This is depicted in Figs. 3, and 4. Fig. 3 (back in Section 3.1) shows the first part of the strategy, where query Q1 is sent to the server, and a superMBR (and corresponding point set) is identified and sent back to the client. Then the query region is updated, which results in query Q2. As we can see, Q2 still resides within the existing superMBR, so its result can be processed locally on the client, and no call to the server is necessary. The next updated query region, Q3, is evaluated on the client. As we can see here, Q3 does not fully reside within the existing superMBR on the client, so Q3 needs to be sent to the server. A new superMBR (shown in Fig. 4) and corresponding point set is found on the server and sent to the client, so that the result for Q3 can now be processed.

One limitation with the First strategy is that a linear search is used to find the clusterMBRs that overlap the query region. Although the set of clusterMBRs is significantly smaller than the set of points, further reducing the cost of the linear search would be ideal. In the Second Strategy, we use a spatial index [3, 13] in place of a linear search, in order to attempt to speed up this part of the server-side operation. The spatial index is used to index the clusterMBRs. The query region is applied directly in the region search strategy employed by the chosen spatial index, in order to identify clusterMBRs that overlap the query region. The remainder of the strategy forming the superMBR, fetching the corresponding points for each chosen clusterMBR, and local processing on the client whenever possible are identical to those in the First Strategy. For our work, we chose the mqrtree spatial index [8] for indexing the clusterMBRs. The mqrtree has been shown to outperform an existing benchmark strategy with respect to region query processing performance when indexing rectangular regions. More information on the mqrtree algorithms can be found in [8]. Fig. 5 shows the mqrtree for the clusterMBRs in Fig. 1.

4. Evaluation

In this section we present the details on the method and results of our evaluation. We compare both strategies with a baseline strategy, in which all queries are automatically sent to the server. We conduct four sets of evaluations: varying both the number of points and their distribution (case 1), varying the size of the region query (case2), varying the clustering algorithm used to initially cluster the points (case 3), and varying the number of clusters formed from

Table 1. Case 1 - Data Transmission Cost

Data Set	Strategy1&2	Baseline
up_500	25,875	72,021
up_1000	34,589	219,961
up_5000	212,259	2,825,459
up_10000	410,451	7,046,400
up_50000	1,907,491	86,354,214
up_100000	5,242,546	241,698,953
ep_500	38,683	64,802
ep_1000	48,962	158,205
ep_5000	232,480	1,756,012
ep_10000	465,641	4,715,655
ep_50000	2,360,457	60,683,301
ep_100000	4,754,561	174,099,942

Table 2. Case 1 - Running Time

Data Set	Strategy1	Strategy2	Baseline
up_500	69.7	206.0	212.9
up_1000	52.7	127.5	345.3
up_5000	221.8	276.9	1308.1
up_10000	519.8	554.3	3042.9
up_50000	4480.4	4386.1	45020.0
up_100000	14645.2	14625.7	156163.6
ep_500	147.1	392.1	210.0
ep_1000	154.0	432.3	336.6
ep_5000	243.8	384.9	1282.6
ep_10000	458.1	517.5	2262.6
ep_50000	3445.5	3358.6b	34732.6
ep_100000	9101.8	9023.1	136944.8

the point set (case 4). For each variation of each evaluation, we execute it 10 times and take the average of the results. We record the following data: 1) the data transmission cost, in bytes (4 bytes for integer, 8 bytes for floating point number, and 1 byte per character); 2) the running time on the server, in milliseconds; and 3) the number of times a request for a new result must be made by the client. Our experiments used simulated point sets. We created several sets of n points, where each set occupies a region of $\sqrt{n} \times \sqrt{n}$. Altogether, we use 18 sets of points. Six sets contains 500, 1,000, 5,000, 10,000, 50,000, and 100,000 points respectively with a random distribution, and six sets with the same number of points respectively and an exponential (i.e., skewed) distribution. The remaining six sets are the query sets, with each corresponding to a specific set size. The number of points in each query set is $\sqrt{n}$ minus the beginning and end points, with the query points proceeding diagonally across the $\sqrt{n} \times \sqrt{n}$ space. Therefore, the number of queries for 500, 1,000, 5,000, 10,000, 50,000, and 100,000 points are 19, 28, 63, 89, 198 and 283 respectively. For clustering, we used both k-means and Expectation-Maximization (EM) [4] from the data mining package WEKA[1], due to its stable implementation of algorithms and our familiarity with the software.

4.1. Results

For case 1, we varied the number of points in the point set and the distribution of the points. We used the k-means clustering algorithm to form 10 clusters for each data set, and a query region size of 10% of the data space. Tables 1 and 2 depict the data transmission and running time costs respectively. We see in Table 1 that the amount of data that must be transmitted between the client and server does increase as the number of points in the point set increases, for both random and exponential distributions. However, the increase is more gradual than that for the baseline strategy, where the increase is between approximately three times and 50 times that produced by the proposed strategies. Next, we see in Table 2 that the running time is similar for both proposed strategies in the larger points sets, and significantly lower than the baseline. For the smaller point sets, we see that the baseline outperforms in some cases, and the First Strategy outperforms in some cases. This is due to the need to re-construct the mqrtree before it can be searched. With a different implementation, the cost for the Second Strategy is expected to be lower. Finally, with respect to the number of queries being sent from the client (not shown due to space limitations), we fount that for the uniform point sets, we have between 3-5 being sent, regardless of the overall number of queries and points. This is due to the choice of a 10% query size and a constant number (10) of clusters (and clusterMBRs). The resulting superMBRs are expected to be larger, and therefore fewer query requests are needed.

For case 2, we varied the query region size between 2.5%, 5%, 7.5% and 10% of the data space. We used the uniform 10,000 point set and k-means clustering algorithm to form 10 clusters. Figs. 3 and 4 depict the data transmission and running time costs of this set of tests. With respect to the number of query requests being made from the

[1] https://www.cs.waikato.ac.nz/ml/weka/

Table 3. Case 2 - Data Transmission Cost

Query Size	Strategy1&2	Baseline
2.5%	361,741	4,843,943
5%	450,139	5,546,478
7.5%	410,451	6,432,801
10%	410,451	7,046,400

Table 4. Case 2 - Running Time

Query Size	Strategy1	Strategy2	Baseline
2.5%	161.3	251.3	1660.5
5%	259.2	321.2	1997.2
7.53%	377.5	411.0	2533.7
10%	526.7	556.5	3024.0

Table 5. Case 3 - Data Transmission Cost

Clustering	Strategy1&2	Baseline
k-means	410,451	7,046,400
EM (Default)	382,997	11,671,734
EM (10 clusters)	521,994	12,161,479

Table 6. Case 3 - Running Time

Clustering	Strategy1	Strategy2	Baseline
k-means	529.8	548.3	3028.4
EM (Default)	302.8	322.8	3689.0
EM (10 clusters)	591.6	621.1	3752.1

Table 7. Case 4 - Data Transmission Cost

#Clusters	Strategy1&2	Baseline
10	410451	7046400
20	292403	4657477
30	326414	3671007
40	297886	3077296
50	279961	2824144
60	271116	2282195
70	264043	2166773
80	247185	2020092
90	255611	1951831
100	256157	1852408

Table 8. Case 4 - Running Time

#Clusters	Strategy1	Strategy2	Baseline
10	510.0	533.6	2947.9
20	526.5	549.6	2757.0
30	687.3	698.2	2490.3
40	694.0	713.3	2391.5
50	684.2	737.5	2394.4
60	772.2	823.9	2267.1
70	715.3	752.0	2247.9
80	686.8	731.4	2256.8
90	721.1	771.3	2276.2
100	679.7	718.1	2165.1

client (again not shown due to space limitations), we found that between 4-5 queries are being issued, regardless of the size of the query. We feel this is related to the results in Fig. 3, which also shows that the amount of data being transmitted by the proposed strategies is not significantly changing, despite the query size. This is due to the constant number of clusterMBRs that exist. Even though the query size is changing, the resulting superMBRs being generated are not, since the setup of the clusterMBRs is constant. In particular, for the 7.5% and 10% cases, both values are the same. Therefore, overlap is taking place between the queries and the same clusterMBRs. Regardless, we still see that both strategies outperform the baseline, which is showing an increase in data transmission costs. Next, we see in Fig. 4 that the running time for all strategies increases as the query size increases. For the Second Strategy, this is occurring due to the mqrtree search strategy a larger query region may overlap more non-leaf MBRs in the index, and therefore processing will take longer. In addition, the cost of building the tree in being incorporated, and with this being removed, it is expected that this cost will be lower than that for the First Strategy. However, it is unclear why the costs are increasing for the First Strategy since the data transmission cost is not increasing and the linear search must be performed across all clusterMBRs. This needs to be further investigated.

For case 3, we vary the clustering strategy. Both *k*-means and Expectation-Maximization (EM) clustering are used. For EM, we use both the "default" setting of 7 clusters, and a manual setting of 10 clusters. The uniformly distributed 10,000 point set is used, and the query size is set to 10% of the data space. Figs. 5 and 6 depict the results of the data transmission and running time costs. We also found that the number of queries being set to the server is between 3-4. We find in both figures that the EM (default) provided the best setting. Since we use WEKA for clustering, it finds the best default clustering using cross-validation. Therefore, it is producing one that results in the lowest data transmission and running time costs.

Finally, for case 4, we vary the number of clusters (and therefore, clusterMBRs) that are created for the point set. We use *k*-means for this, the uniform 10,000-point set, and a query size of 10% of the data space. Figs. 7 and 8 shows the final results. We observe some interesting results. First, in both cases, as the number of clusters increases, the running time and data transmission costs decreases for the baseline strategy. This is a result of the following. As the number of clusterMBRs increase, their size decreases, and therefore the resulting superMBRs decrease in size as well. With respect to the proposed strategies, the running times and data transmission costs fluctuate up and down. This is due to the variations in the superMBRs that are generated, given the increase in the number of clusterMBRs and variations in the size of the resulting superMBRs.

5. Conclusion

We propose two continuous region query processing strategies over a clustered point set. On the server, both find the appropriate clusterMBRs, determine a safe region and send it with the corresponding point data to the user. On the client, each time the user wants to issue an updated query, the query is checked against the superMBR in order to avoid unnecessary server requests. We also compare different searching strategies - linear and indexed. An evaluation versus a baseline shows noticeable improvements in the server workload (i.e., the number of queries sent to the server), data communication cost, and query running time in our first and second strategies over the baseline strategy. The data communication cost and server workload of our first strategy are the same as our second strategy. We also found out that our second strategy has better performance on query running time over our first strategy in bigger data sets. In the future, we wish to evaluate our strategy versus other strategies, and with other clustering strategies. We will also study moving region queries over moving data points and spatial network data, since these types of queries has many applications in location-based services. In addition, we can expand our work into k-nearest neighbor (kNN) queries.

References

[1] Al-Khalidi, H., Taniar, D., Betts, J., Alamri, S., 2013. Efficient monitoring of moving mobile device range queries using dynamic safe regions. In: Proceedings of the 11th International Conference Advances in Mobile Computing and Multimedia. pp. 351–360.
[2] Al-Khalidi, H., Taniar, D., Betts, J., Alamri, S., 2013. On finding safe regions for moving range queries. Mathematical and Computer Modelling 58 (56), 1449–1458.
[3] Gaede, V., Günther, O., 1998. Multidimensional access methods. ACM Computing Surveys 30, 170–231.
[4] Han, J., Kamber, M., Pei, J., 2011. Data Mining: Concepts and Techniques. Morgan Kaufmann.
[5] Hu, H., Xu, J., Wong, W., Zheng, B., Lee, D., Lee, W.-C., 2005. Proactive caching for spatial queries in mobile environments. In: Proc. 21st Int't Conf. on Data Engineering.
[6] Jung, H., Kang, S.-W., Song, M., Im, S., Kim, J., Hwang, C.-S., 2006. Towards real-time processing of monitoring continuous k-nearest neighbour queries. In: Proc. 2006 Int'l Conf. Frontiers of High Performance Computing and Networking.
[7] Lee, K., Lee, W.-C., Leong, H., Unger, B., Zhang, B., 2009. 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.
[8] Moreau, M., Osborn, W., 2012. mqr-tree: a two-dimensional spatial access method. Journal for Computer Science and Engineering 15.
[9] Osborn, W., 2016. Continuous region query processing in the mqr-tree. In: Proceedings of the 25th International Conference on Software Engineering and Data Engineering.
[10] Park, K., M., S., Hwang, C.-S., 2005. Location-based services for dynamic range queries. Journal of Communications and Networks 7 (4), 478–488.
[11] Park, Y., Bok, K., Yoo, J., 2012. An efficient path nearest neighbour query processing scheme for location-based services. In: Proc. 17th Int'l Conf. Database Systems for Advanced Applications.
[12] Schiller, J. H., Voisard, A. (Eds.), 2004. Location-Based Services. Morgan Kaufmann.
[13] Shekhar, S., Chawla, S., 2003. Spatial Databases: A Tour. Prentice Hall.
[14] Tao, Y., Papadias, D., Shen, Q., 2002. Continuous nearest neighbor search. In: Proc. 28th Int'l Conf. Very Large Data Bases. pp. 287–298.
[15] Wu, K.-L., Chen, S.-K., Yu, P., 2005. Efficient processing of continual range queries for location-aware mobile services. Information Systems Frontiers 7 (4-5), 435–448.
[16] Zhang, J., Zhu, M., Papadias, D., Tao, Y., Lee, D., 2003. Location-based spatial queries. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. pp. 443–454.