CONTINUOUS SPATIAL QUERY PROCESSING OVER CLUSTERED DATA SET

FARZIN KEYKAVOOS AMAND
Bachelor of Engineering, Seraj University, 2011

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

CONTINUOUS SPATIAL QUERY PROCESSING OVER CLUSTERED DATA SET

FARZIN KEYKAVOOS AMAND

Date of Defence: December 10, 2018

Dr. W. Osborn Supervisor	Associate Professor	Ph.D.
Dr. Y. Chali Committee Member	Professor	Ph.D.
Dr. J. Anvik Committee Member	Assistant Professor	Ph.D.
Dr. H. Cheng Chair, Thesis Examination Committee	Associate Professor	Ph.D.

Dedication

To my beloved mother.

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 moving (continuous) region query in which a moving user continuously sends queries to a central server to obtain data or information. In this thesis, we introduce two strategies to process a spatial moving query over clustered data sets. Both strategies utilize a validity 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, response time, and workload costs.

Acknowledgments

First, I would like to express my most profound gratitude to my supervisor Dr. Wendy Osborn for giving me this opportunity to pursue my studies in the masters program. This thesis work could not be possible without her invaluable help, guidance, support and patience.

Next, I would like to thank my supervisory committee members, Dr. Yllias Chali and Dr. John Anvik for their helpful comments and suggestions.

I also would like to express my deepest gratitude to my sister, Rozita, for her endless support. She is such a wonderful person that I always refer to her when I encounter difficulties, and she kindly helps me and gives me the motivation to go through hard times.

Furthermore, I would like to thank my parents for always giving me the passion and courage for working on my goals.

Last but not least, I would like to thank my friends who make life more easier including Koorosh, Hossein, Behnam, Soroush, and Sarah.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The ever-increasing growth of using mobile devices and advances in wireless communication technologies have led to massive studies in the field of mobile computing [15]. One of the major topics in this field is continuous spatial query processing. A moving client/user continuously sends queries to a central server to obtain data or information based on their current location. Continuous spatial queries need constant updates to the result as the query object or points of interest (POIs) are moving. There are several types of continuous spatial queries [15]. One such type is called a window (or range or region) query. A continuous region query finds the POIs that are within the user-specified region and then returns them back to the user as the query result.

1.1 Motivation

Nowadays, many mobile applications provide location-based services (LBSs) [15] including in navigation, location-based social networks, and mobile gaming. Location-based services support different types of location-dependent queries. One of the popular LBSs is continuous or moving range query over stationary data objects. An example of a range query is : *a tourist continuously asks for Japanese restaurants within a range of 1 Km from his/her location*. As tourist moves, the result of the query will change and hence it needs continuous updates.

Processing a moving region query is challenging as the server is required to continuously compute and send the results to the query issuer. The important factors in an efficient

query processing are as follows:

- Maintain up-to-date results at a low cost (low data communication cost between server and client).
- Keep query response time as low as possible (server workload and search process are crucial for this factor).

Many research works have been carried out in the area of continuous query processing. Zhang et al. [36] present a location-based query approach that enables moving users to determine if their continuous query is still valid from their current location without having to send a new request to the server. To accomplish this, the server returns a validity region along with the query result. Al-Khalidi et al. [3] propose an efficient technique called the range safe region to process a moving query. Due to the irregular shape of the safe region, they use Monte-Carlo simulation to calculate the area of it. In this approach, the client only issues a new query to the sever if it leaves the specified region. Park et al. [22] propose a broadcast-based data delivery scheme (BBS) for continuous region queries. In this scheme, the data objects which are sent by the server are sorted continuously on the server based on their locations. Also, the server sends an index segment along with these data. Yung et al. [35] propose a moving region query processing on road networks. The authors use an approach called safe-exit (a region that the result of the query remains unchanged until the client reaches any exit) to reduce data communication between server and clients. Huang et al. [14] propose a proxy-based technique for continuous window and nearest neighbor (NN) queries. Estimated valid regions (EVRs) are created by the proxy for the client in order to reduce the data transmission cost. They provide the EVR in the shape of vectors, namely estimated window vector (EWV) to create larger valid regions for window queries. We discuss more research work that has been devoted to continuous query processing in Chapter 2.

Most of the existing works in the continuous query processing use a safe region (i.e., validity region, safe-exit, EVR) approach in order to reduce the data communication between

server and clients. For this reason, we use a safe region in our first and second strategies in the form of a superMBR (super minimum bounding rectangle). In addition, most of the existing works on moving region query over static point data do not use clustering techniques or a spatial access method. The ones that are using spatial access method (e.g., R-tree [10]) have high overlap and overcoverage. Hence, we use two different clustering algorithms (i.e., k-means and EM). By clustering our data sets, we improve the search for data on the server. We also utilize a 2-dimensional index structure, the mqr-tree [1] [21], that needs a lower number of disk accesses than other benchmark strategies when doing region search.

1.2 Outline of thesis work

We introduce two region query processing strategies over static point sets which are clustered. In our setting, a client continuously sends region queries to a central server and the server processes the queries and sends the results back to the client. After obtaining the first result (which contains a point data set and safe region), the client checks whether the next query is completely inside the safe region. If it is within the safe region, the client does not need to send the query to the server. Otherwise, the client sends the query to the server to get a new result. The purpose of the safe region method is to:

1. Avoid communication cost between client and server while the query remains in the safe region.

2. Decrease the need for persistence monitoring of the query.

In our first strategy, we use a linear search for finding the overlapping cluster Minimum Bounding Rectangles (MBRs). We also use the safe region approach on the client side to reduce server workload, data transmission cost and query response time. In our second strategy, we utilize the mqr-tree [21] region search on the server to speed up the search process. We also use the safe region approach in this strategy as well. We also present

[1] Marc's quadrant R-tree

a baseline strategy for evaluation purposes. We eliminate the safe region method and the client sends each query to the server without any comparison. Evaluation results show that our first and second strategies that use the safe region method on the client-side significantly outperform our baseline strategy by issuing a lower number of queries to the server which consequently improves the efficiency of the query.

1.3 Contributions

Our work contributes to the continuous spatial query processing research field as follows:

- Using a spatial indexing structure, the mqr-tree, on the server to find the overlapping cluster MBRs. We use the mqr-tree region search to speed-up the search process. The mqr-tree is a 2-dimensional index mechanism that organizes objects (e.g., point, line, and polygon) in a 2-dimensional node in regard to their spatial relationship. This structure lends itself to applications such as region searching.

- We utilize a safe region (i.e., superMBR) to reduce server workload and data transmission cost. A safe region is a region which the movement of the client (i.e., query) within that area makes no changes to the query result. We delegate the safe region to the client to reduce the communication overhead between the server and the client.

- We cluster our data sets and then compare the query with the cluster MBRs which reduces the search process on the server.

1.4 Thesis Overview

The structure of the rest of this thesis is as follows.

In **Chapter 2**, we provide a literature review of the preceding works that have been done on two types of moving query processing (region query and kNN query). Also, we describe some of the privacy-related works in location-based services.

In **Chapter 3**, first, we give a short overview of the clustering algorithms that we used in our work (k-means and EM). Then, we summarize work on the mqr-tree, a 2-dimensional indexing method, and its region search. Lastly, we explain our moving region query processing strategies in detail.

In **Chapter 4**, we describe the settings that we used in our experiment and the metrics to evaluate the performance of our strategies. Furthermore, we present the evaluation results for four different scenarios.

Finally, in **Chapter 5**, we discuss the conclusion of the thesis and present the future works in this research direction.

Chapter 2

Background

In this chapter, the processing of moving queries will be reviewed from the existing literature. It can be classified under three main categories:

1. Moving Window Queries
2. Moving Nearest Neighbor Queries
3. Privacy-related issues in LBS

Our proposed strategies reside in the first category (Moving Window Queries). However, we present the other two categories of moving query processing for completeness of our work.

2.1 Moving Window Queries

Moving window queries involve finding interesting objects inside a definite window (i.e., range, region or radius). However, research covers a far wider scale, which is from finding interesting objects to forming the region containing them [29]. Before starting to work on processing and optimizing range queries, it is crucial to familiarize with the potential of region queries.

Generally, the region queries can be classified in three categories [29]:

1. Finding interesting objects
2. Forming a region that contains the objects

3. Defining centroids for the region

Objects of interest are usually the query results that are inside a unique region, while the query point is the location where the query is issued and the results are corresponding to location of the query and the region defined around it [29].
The main goals of moving query processing are to minimize [35]:

- Computation cost on the server
- Computation overhead of the client
- Communication cost between the client and the server

2.1.1 Safe-Exit Approach

Yung *et al.* [35] propose an approach called *safe exit*, which is a safe region-based method on a road network. A safe exit obtains the border of a safe region, and it has a smaller size. The calculation of safe exits not only needs the result set but also non-result set which adds more complexity to finding the safe exits. A safe region is a region where the query result will not change while the user remains in it. Therefore, this approach results in a huge decline in the communication cost between the client and the server.

According to the authors, safe exits are small in area, which provide a low communication cost between the server and the client. Also, this approach enables the client to check the correctness of the query result locally, and in a timely manner. When a client leaves a safe exit, it needs to send a request to the server again to re-compute both the query result and the safe exit.
The safe exit computation on the server-side consists of two steps [35]:

1. Retrieving a sufficient set of points, which can contribute to the formation of the safe region.

2. Computing the safe region by using the retrieved points. First, each retrieved point is added to the current safe region. Then, it is narrowed down to the smallest region possible.

The measurements that the authors used for assessing the performance of their proposed solution are [35]:

1. The total communication cost.
2. The total communication frequency.
3. The total CPU time of the client and the server sides.

The authors report in some experiments that the USA road network (175k nodes, 179k edges) is used. They show that the proposed pruning rules significantly reduce the number of data points and network nodes that need to be processed. In addition, different ordering heuristic methods are applied to compute safe exits rapidly.

2.1.2 Finding Safe Regions

AL-Khalidi *et al.* [3] use Monte-Carlo simulation for creating safe regions to reduce the communication cost when processing moving range queries. The authors state that the common approach for determining a safe region is a Voronoi Diagram (VD) [9]. A Voronoi Diagram partitions a plane into sections called Voronoi cells. For a given set of spatial objects(i.e., sites, or seeds), each Voronoi cell specifies points that are closest to a site. However, the authors identify some disadvantages, such as requiring continuous updating, which is very costly. Therefore, Monte-Carlo simulation can be used to calculate the safe region efficiently.

The authors define a safe region as an irregular shape when there are more than two objects in it. For Monte-Carlo simulation the queries are determined using a bounding region with a known size. Multiple points are randomly generated within the bounding

region, and the number falling within the safe region are totaled.

The Safe Region Area (SRA) is measured as follows:

$$SRA = \frac{\text{sum of points in safe region}}{\text{total random points}} \times \text{area of bounding region}$$

The purpose of this method is to [3]:

1. Avoid communication costs while the query is inside the safe region
2. Decrease the need for persistence monitoring of the query
3. Rule out the need to follow a pre-defined path

The authors use two performance measures to evaluate their strategy:

1. Comparing the average size of the safe region to the size of the data space
2. Counting the number of safe regions that are passed over when the query moves inside the data space

The results show that their proposed methods reduce the monitoring of queries and provide more privacy for users whenever they move inside the safe region.

2.1.3 COVET Query Index

Wu *et al.* [30] proposed a memory-based approach that uses a query index to evaluate continuous range queries. Their covering tile-based (COVET) query index is a set of predefined virtual tiles, where one or more are used to precisely cover the region which is defined by a range query. It enables evaluation of the query to benefit from the incremental changes in the locations of the object. It is also stated [30] that, since the movement of range queries are less frequent than the movement of objects, it will be more efficient to create an index on the range queries.

The effects of size and shape of the tiles are also examined by the authors. It is found that a not-too-small and not-too-big tile size is best, but the optimum tile size ultimately

depends on the query size distribution. It is also found that, since square tiles require less storage, they are more effective than rectangular tiles, while they achieve comparable results. In conclusion, the authors found that, in overall re-evaluation time the COVET index works better than a cell-based index since it benefits from the incremental changes in object locations.

2.1.4 Dynamic Safe Regions

AL-Khalidi *et al.* [2] propose two approaches that address a problem with a safe region in that identifying exactly when and where the user will leave that safe region is typically a major problem. The approaches are [2]:

1. Re-construct the safe region by using only the two objects closest to the boundary of the moving client

2. Periodically monitor the position of the client

The authors achieve this by using a linear model to monitor the moving range query inside the safe region. Basically, by this method they could predict when the query will leave the safe region based on its current location and velocity. By using this technique, not only is the need for continuous monitoring of the query reduced, but also the need for the user to follow a defined path is eliminated. The evaluations show that this method expands the safe region compared to other approaches in order to reduce the wireless communication and query re-evaluation costs to maintain up-to-date query results.

2.1.5 Verifying continuous range query

Yung *et al.* [34] proposed an efficient technique to verify the correctness of the safe regions of moving circular range queries. The authors present a new notion called verification objects (VO)-optimal to illustrate verification approaches that obtain the minimum data points and the Merkle R-tree (MR-tree) entries for the VO. A MR-tree is a data structure

based on Merkle tree [20] and R*-tree [5] for authenticating the safe regions on the client-side. Their reason for using the MR-tree for VO-optimality is that it is popular and has low communication cost. The VO is proposed by the authors to authenticate both the safe regions and the results of continuous range queries. This proposed technique is reported to also optimize the communication cost which enables the client to update the safe region without contacting the location base server even when the client is not in the safe region.

An arc-based approach is used to construct the VO. The idea of this approach is to use the arcs of a safe region to create a VO with the smallest size. The arc-based method includes two algorithms for both the server and the client. For the server algorithm, the necessary data for authenticating the range query result and the safe region are put into VO and then broadcasted to the client. For the client algorithm, the accuracy of the safe region and the result query is confirmed according to the data saved in the VO.

The authors justified that their methods for verification of continuous range queries can minimize the broadcasted data between the server and the clients. Their experiments show that their methods effectively verified the continuous range queries while the communication costs and additional computations are minimized.

2.1.6 Dynamic range queries in LBSs

Park *et al.* [22] presented a broadcast-based data delivery scheme (BBS) for moving range queries. In their scheme, the data objects which are transmitted by the server are sorted continuously on the server based on their locations. Also, the server sends an index segment along with these data. However, the user does not need to wait for an index segment if the required data objects have already been identified before the index segment has arrived.

The BBS method tries to make the access latency lower for the user. Also, it can reduce both the query answering time and tuning time. By using the BBS, the user can issue range queries without any need to listen to the broadcast station, when the required data objects

are previously pre-fetched into the cache system.

Two aspects are considered in the indexing approach [22]:

1. Access latency: The average time when a user sends a query to the server and the moment when the result query received by the user

2. Tuning time: The amount of time listening to the broadcast channel by a user

The experimental results demonstrated that the BBS method has lower access latency time than other spatial index schemes, and that the user does not always need the index segment.

2.2 Moving Nearest Neighbor Queries

In this section, we present a summary of classifications and individual strategies of nearest neighbor (NN) queries. Taniar *et al.*[28] summarizes several ways to classify nearest neighbor (NN) query strategies. First, NN queries are classified in four perspectives:

1. Space perspective: The queries can use Euclidean space, spatial road networks space, or both

2. Result perspective: The query result can be more than k objects and also add further information (e.g., path data)

3. Query-point perspective: There is a chance of getting a single or multiple points for the query

4. Relational perspective: It clarifies the relation between the query point and the target object

There are two types of spatial road network algorithms, which are classified as follows [28]:

1. Based on the road network, as a graph form (graph-based kNN)

2. Based on the Network Voronoi Diagram (NVD-kNN)

Further, kNN queries can be processed on spatial road networks by the following approaches [28]:

1. Restriction: This is a top-down approach, which builds an upper boundary for the search area, and then during the process, it prunes the search area. This process continues until the nearest neighbors are found. A spatial index [28] can be used for this purpose. For example, the key point in using spatial indexing is to start from a large search space, and then reduce it to a smaller area.

2. Expansion: This approach is bottom-up, starting from the query point and progressing to acquire the wanted objects. It expands one segment in each time period. Both the NVD-kNN and the graph-based classes of kNN query processing algorithms use expansion. The NVD-based method expands with the Voronoi polygon while the graph-based method expands with the road segment.

Taniar et al. [28] also identify two categories in kNN query processing that are not dependent on the k value. The first is where the range and kNN constraints are mixed. The second is in approximated distance between objects in the query result.

The query points are at different locations in some kinds of kNN queries [28]:

1. Considering all existing objects as query points (all-kNN (akNN) query)

2. Selected objects considered as the query points (Group Nearest Neighbor (GNN) query)

3. A GNN in another form (Group Nearest Group (GNG) query)

4. Just objects in a certain range or region (Range-kNN)

There are several methods for processing a GNN query, where some are as follows [28]:

1. Multiple query: First, it finds the 1NN of each query point and computes the distance towards the rest of the query points. Then, it obtains a threshold distance by the

minimum sum of the distances. Then, it finds the next NN by comparing the total and the threshold distances. It halts when the NN has a further distance than the threshold.

2. Single point: Sometimes, the target object may be the same as the query point. To solve this, the single point approach uses a single traversal of the GNN query.

3. Minimum bounding box: It is the same as the single point method. The difference is that the Minimum bounding box method uses a bounding box instead of centroid to cover the query points.

4. Group closest pairs: It looks for the nearest pairs of objects from all existing objects.

There are two relationships between a query point and its nearest neighbor [28]:

1. Bi-directional: Target object is NN of query point, while the query point is the NN of the target object.

2. Reversed: Instead of determining which objects are the NN to a query point, the target objects determine which one is NN to a query point.

2.2.1 Authentication of Moving kNN Queries

Yung *et al.* [33] also proposed a framework to authenticate the results of safe regions for moving kNN queries. In their MR-tree approach, digests of nodes are continuously evaluated from the leaf to the root level. Then, by signing the root digest using the private key of the data owner, the signature of the dataset is obtained.

For moving kNN queries, the MR-tree approach utilizes an order-k Voronoi cell as the safe region for moving kNN queries. In every given time unit T, the client requests a kNN query from the server. Then, the correctness of the results is verified by an MR-tree.

The authors proposed two strategies for processing using the authentication of moving kNN Queries [33]:

1. Vertex

2. Materialization

The vertex-based approach (VA) verifies a moving kNN query by using a compact set of Verification Objects (VO) and the relationship between the kNN result and the vertices of an order-k Voronoi cell. The server computes the kNN for a client query from an MR-tree and a safe region, which are verified using a verification region.

The materialization approach (MA) attempts to reduce the computational cost on both the server and the client sides [32]. It uses a Voronoi MR-tree, which is an extension of the VoR-tree (The VoR-tree [25] is an R-tree [10] in which each leaf entry saves both a point and a generator set). For a client query, the kNN result is computed from a Voronoi MR-tree by the server. Then, it defines a verification region to identify points for the client to verify the kNN result and the safe regions. Unlike the VA approach, the verification region includes just one part, which is the common kNN verification region. The client continues to authenticate the correctness of the kNN result and the safe region defined by the VO.

According to the authors, the advantage of the VA approach is a low communication cost per query. However, on both the user and the server sides, the online calculation of order-k Voronoi cell at run-time is needed. The VA approach is found to have an efficient communication, while MA is found to have lower computational cost.

2.2.2 COMET

Cho *et al.* [6] propose a collaborative strategy called COMET to process moving kNN queries (MkNN) in directed and dynamic road networks. It minimizes the communication and computational costs by passing a portion of the processing to the query object. When a query object goes into a new active segment, or the traffic status around it has changed, it sends its location to the server. An influence region then is presented which tracks changes in the safe segment.

In their approach, the processing of MkNN queries is performed by the communication

of the server with the query objects. The interaction between the server and query object is called the matching of the query request and response. Each time the query object goes into a new road segment, the interaction is repeated. When the changing of traffic condition in the neighborhood of the query object does not satisfy its safe segment, the server initiates a proactive interaction between the query object and the server. For various workloads, experimental results show that COMET performs better than a conventional method in terms of query processing time, energy consumption and bandwidth usage.

2.2.3 Island-Based and Adaptive Island-Based Algorithms

Kim and Chang [17] proposed two new algorithms for kNN query processing using a query region in road networks. Both IB-kRNN (Island Based K-Range NN) and an adaptive IB-kRNN (AIB-kRNN) utilize Island indices to reduce the number of points of interest that are retrieved. IB-kRNN spreads the Island scheme to the distances between the nodes and the nearest points of interest (POI). Each node has an Island index that saves the pre-calculated distances to all POIs which are placed inside a definite radius from the node.
The authors state that the advantages of IB-kRNN algorithm are as follow [17]:

1. Reducing the number of network expansions by pre-evaluating the distances between nodes and their adjacent POIs.

2. Processing a kNN query over a query region very quickly by utilizing the Island index which resides in main memory.

The experimental results illustrated in this paper show that both algorithms have better performance than the k-Range NN [4] in terms of the amount of retrieved POIs and the query processing time.

2.2.4 Inverted Grid Index

Ji *et al.* [16] studied the process of performing NN queries in distributed surroundings. The authors proposed a distributed Inverted Grid Index, which is a mixture of an inverted

index and a grid partition, in multi-dimensional space. Their grid-based index is simple, so it can be efficiently updated and disseminated. The Inverted Grid Index lists the objects per cell. They utilize MapReduce [8] for the implementation of the mapper and reducer with no changes being performed on the MapReduce framework. As the user submits a new NN query, the computation module retrieves the query results.

The authors used a combination of offline pre-evaluation and online query processing. MapReduce re-computes the Inverted Grid Index so that an NN query can be processed quickly whenever needed. When the dataset is changed, the index should be rebuilt by running the offline module.

The experimental results show that as the number of cluster nodes increases, the construction time of the index structure decreases linearly. Also, the results show the good efficiency and scalability of NN queries based on the Inverted Grid Index.

2.2.5 Influential Neighbor Sets

Li *et al.* [18] proposed a new approach which uses a small set of safe guarding objects, instead of a safe region, to process moving kNN (MkNN) queries. The idea of their approach is that, there is no need to recompute the query result as long as the current kNNs are closer to the query object than the safe guarding objects. The safe guarding objects set the region to the largest possible safe region which minimize the high recomputation costs.

As the query object moves continuously, the current kNN objects will be replaced by safe guarding objects which are closer to the current kNN. According to the authors, no safe guarding objects can becomes a new kNN, the current kNN set stays valid and so no recomputation is needed. The authors also defined a special set of safe guarding objects, called an influential neighbor set, for efficient computation. The main reason to propose the influential set is to ensure the validity of the kNN set.

The authors compared their method with a state-of-the-art algorithm, and the results show that it has better performance in I/O and computational costs.

2.2.6 Proxy-Based Scheme

Huang *et al.* [14] proposed a proxy-based method for a moving NN. The proxy constructs estimated valid regions (EVRs) for moving users by utilizing the spatiotemporal domain of spatial queries. For NN queries, they developed two algorithms to speed up EVR creation which leads to establish adequate EVRs by the proxy even if the size of the cache is small.

In addition, they used an index structure called EVR-tree for NN. To increase productivity, the authors developed algorithms to achieve the results of NN queries and to help with grid index growth. The grid index is applied to ensure efficient NN query response and the updating of the EVR.

The authors conducted different experiments to evaluate the performance of proposed algorithms. The results showed that their approach has the significant advantage towards the other proxy-based methods. It is concluded that their proxy-based method is sufficient in a condense populated space, while the server-based approach is sufficient as users move at high speed.

2.3 Privacy-related issues in LBS

In the recent years the number of mobile users [24] has drastically increased. These users can access the information they need regardless of the user locations through mobile networks. Mobile information services provide several types of information in the moving environment. However, due to the large number of mobile users, the issue of user privacy has arisen [24]. In the process of accessing the mobile information services, a mobile user sends personal information to the service provider, which puts the privacy of the user at risk, since their information could be easily accessed by other parties. Therefore, this privacy issue has created a real demand from the mobile users to expect more dependable mobile information services that protect their privacy.

Privacy protection is an important factor in location-dependent systems. Location in-

formation is a significant topic in privacy of spatial queries, when a user issues a query. So, in this section the privacy-related techniques and algorithms are reviewed.

2.3.1 Casper: a privacy-conscious query processor

Chow *et al.* [7] propose a privacy-aware query processing model called Casper, which enables the user to obtain continuous location-based services without disclosing their actual location. A privacy-aware query processor is located at the server to handle continuous queries from the cloaked location of the user. It does not need to know how the cloaked location is determined. It also supports three new privacy-aware query types, which provide a trade-off between query processing cost and answer optimality [7]

1. private queries over public data
2. public queries over private data
3. private queries over private data

A shared execution model is proposed by the authors to enhance the system scalability of processing continuous privacy-aware queries. A set of selected static continuous query results are maintained and shared by all outstanding continuous queries.

The location anonymizer is a middle layer between the mobile user and the server which alters the query location and the user identity information and sends a cloaked area to the server. It also uses a candidate list of answers from the server to compute the exact answer and send it to the user.

The authors state four techniques for location anonymization [7]:

1. False dummies: when updating their location, the user sends different locations to the server, for which only one of them is true.
2. Landmark objects: in this approach, the user sends a definite landmark or object.

3. Location perturbation: the idea is to make the user location hazy by using temporal or spatial cloaking.

4. Avoid location tracking: this approach is different from the three previous approaches because it does not hide any locations. Instead it tries to avoid the tracking of user activities.

In the Casper method, location perturbation is used, due to less computational cost and more adequate query processing.

Privacy-aware query processing is divided into three major classes [7]:

1. Location obstruction: the user sends a false location to the server, and then the server continuously sends the list of nearest objects to that false location until it satisfies the user privacy.

2. Space transformation: the initial location of the query and data is transformed into another space by using a secure third party.

3. Cloaked area processing: a privacy-aware query processor is installed on the server to manage the cloaked spatial area obtained from the both user and the secure third party. The authors use this approach to perform privacy-aware query processing.

The authors give experimental results which show that their proposed privacy-aware query processor acquires a high quality snapshot and continuous location based services, while protecting the user's location privacy.

2.3.2 An algorithm to stand against overlapping rectangle attack

A way to protect user's privacy (i.e., location) is to send a region instead of user's exact location [13]. However, when a user is continuously sending queries (i.e., moving), a location based service provider (LSP) can refine the location of the user by considering the overlap of successive rectangles. This is a thread to user privacy and called the overlapping rectangle attack.

Hashem *et al.* [13] studied the ways to protect user privacy information (e.g., location) from an overlapping rectangle attack in a moving kNN (MkNN) query as the private moving kNN (PMkNN) query. In this method, users can determine an accuracy of the query answers by adding an extra x amount to nearest data objects (i.e., k + x) where x is usually a small integer number. Furthermore, the user sends his inaccurate location as a rectangle, called an obfuscation rectangle.

The authors introduce two solutions to encounter the overlapping rectangle attack:

1. Ignoring service quality (e.g., less accuracy of answers)

2. Obtaining some additional data from the server

They propose a new algorithm, called CLAPPING (Confidence Level Aware Privacy Protection in NN queries), which finds the kNNs with a defined confidence level (a factor that assures the distances of returned objects to the user's location are within a definite portion of the real kth NN's distance) for an obfuscation rectangle. The experimental results show that it has faster performance than Casper [7] and also lower I/Os times.

2.3.3 Privacy-aware location dependent queries

Xu *et al.* [31] propose a location cloaking method which confronts against a trace analysis attack. Two goals are targeted in the cloaking process:

1. Improving the cloaking quality during a trace analysis attack

2. Keeping the size of result query as small as possible

Three new solutions are identified by the authors about the location cloaking approach to protect location privacy. First, cloaking regions are presented. It is shown that by using a circle shape for the cloaking region in region queries, a smaller query result is created than using other shapes. Second, a mobile cloaking method is developed to stand against a trace analysis attack. Lastly, a polynomial algorithm is created for calculating circular range-based kNN queries on the server.

The authors design two algorithms, called MaxAccu and MinComm, for cloaking the exact location of users. MaxAccu makes high query accuracy achievable, while MinComm tries to minimize the communication cost of the network and provide efficient accuracy as well.

For returning query results, two methods, called bulk and progressive, are proposed which give results in both cumulative and accumulative ways. For large result supersets, the progressive approach obtains a shorter response time of performing evaluation of the query and sending the result at the same time.

There are two major methods to protect users privacy in location based environments [31]:

1. Trusted on the server to minimize the access to location of the user by some pre-defined policies

2. An agent works between the user and the server. It blurs users real location before sending it to server.

Experimental results show that their proposed cloaking algorithms clearly enhanced the quality of location cloaking based on entropy measures.

2.3.4 Range-kNN Algorithm

An issue with both kNN and range queries is that they are highly dependent on their location that is provided by the location based services (LBSs). However, sometimes the LBS fails to provide the accurate location.

Based on this identified drawback, Shao and Taniar [23] proposed Range-kNN queries in order to solve the problem of inaccurate location. Their proposed algorithm can accept an irregular shape as the query range from the user, and then the query result is provided according to distances between the desired objects and the query range. The Range-kNN is defined based on these two aspects:

1. All desired objects are regarded as 1NN if they are inside the query range or intersections.

2. All other interesting objects are iNN if they are outside the query range.

For processing Range-kNN queries, they divided the approach into three major steps [23]:

1. Transformation of query range: The main aim of this optional step is to change the irregular shape with an irregular polygon.

2. Defining distances between objects and the query polygon: to compare the distances between objects and the query range, these distances are defined.

3. Search Algorithm: After having the defined distances in this last step, an R-tree index is used to avoid the linear search.

The authors concluded that their proposed algorithm promises to improve the accuracy of processing results without adding any significant computational overhead.

2.3.5 Secure Top-k query processing through Untrusted LBS

Zhang *et al.* [37] proposed three patterns to identify an incorrect spatial snapshot and continuous top-k query results. Their system has three main components:

1. a data collector: collects reviews about POIs

2. data contributors: common people who send reviews about POIs

3. location based service provider (LBSP): buy POIs from data collector, and enable users to request spatial top-k queries. However, LBSs are not trustable and may provide wrong query results to users for different reasons.

The main objective of their patterns is that a data collector produces some additional information about its data set, and sells them to LBSs along with the data set. In addition, it gives the user the chance to authenticate the correctness of the query result which is received from the LBSP. To verify this correctness, all k POIs should be in data collector and the top-k POI in the query region.

The LBS responds to a top-k query by returning the top-k POI along with proofs for correctness and validity which is created from authenticated hints. The validity proof enables the user to determine that the query result includes authenticated data from reliable data collectors. The correctness proof allows the user to justify that the received top-k POIs are satisfying.

The first two patterns authenticate and pre-evaluate snapshot top-k queries but in different ways. The third pattern, which is based on the first pattern, detects valuable and confirmable continuous top-k queries. The authors studied the usefulness and efficiency of these patterns via precise simulations.

2.3.6 Range-kNN queries in a moving environment

Shao *et al.* [24] proposed an innovative algorithm to process Range-kNN queries, which utilizes a region (i.e. query range) instead of one single query point to secure users privacy. In this proposed approach, the authors divided the algorithm into three main steps:

1. Landmark tree creation: This first step is based on a new concept which they called the Landmark Tree (LT). The LT is used to index all the location information in a multi-level tree by dividing the map area into a number of landmarks, to be able to hide the exact location of the user in a definite radius. With this LT approach, a query range is sent to the server instead of the exact location of the user.

2. Query range definition: After the creation of the LT which is constructed from the real map, the query range definition is able to find a sufficient landmark to protect itself.

3. Search algorithm: After the definition of the query range, the algorithm performs nearest neighbor query processing, which are categorized into two kinds, the ones that are inside the query range as 1NN, and the others which are outside of the query range as iNN. For handling this search process, the authors proposed a Voronoi-based algorithm.

In this work, the accuracy of Range-kNN queries is also compared to the range queries. It is concluded that this proposed algorithm is solidly more accurate than range queries for moving users. Then the performance of Network Voronoi Diagram re-generation process is evaluated in comparison with the overall performance of the search algorithm, which shows this proposed algorithm is better at dealing with Range-kNN queries in different objects densities. Finally, although this algorithm is proven to be efficient in processing Range-kNN queries, it only focuses on the static objects.

2.4 Conclusion

In this chapter, we reviewed the existing research works in the moving query processing. In the next chapter, we will explain our proposed strategies for continuous region query processing.

Chapter 3

Continuous Region Query Processing Strategies

In this chapter, we introduce two strategies for continuous region query processing over clustered data sets. In the first strategy, we use a linear search algorithm to find the cluster MBRs (cluster minimum bounding rectangles) that overlap with the query on the server, and utilize a superMBR (super minimum bounding rectangle) technique on the client-side to reduce the server workload and data transmission cost. For the second strategy, we use a spatial indexing mechanism, the mqr-tree [2] [21] instead of the linear search algorithm to attempt to speed up the search process especially when we have a large number of clusters. In both strategies, a user continuously sends region queries to the server when required, and the server computes and sends the result back to the user. In the baseline strategy, we send all queries to the server without any comparison of the queries with the superMBR region that we use in our first two strategies.

The rest of this chapter is organized as follows. In Section 3.1, we present a brief description about the clustering algorithms that are used in our work. In Section 3.2, we introduce the mqr-tree spatial indexing method and describe how its region search works. In Section 3.3, we describe our first strategy and give an example about how the server finds the overlapping cluster MBRs and determines the superMBR. In Section 3.4, we describe our second strategy that uses the mqr-tree region search on the server to speed up the search process. Finally, in Section 3.5, we describe our baseline strategy which directly sends all

[2] Marc's quadrant R-tree

of the queries to the server without any comparison with the superMBR on the client side.

3.1 Clustering Methods

Clustering is one of the data mining tools that is applied in many fields such as, business, biology and spatial database technology [12]. In this section, we introduce the clustering algorithms that we use in our work. In section 3.1.1, k-means algorithm is discussed. In section 3.1.2 a brief description of EM algorithm is presented.

3.1.1 k-means

The k-means algorithm [12] is a partitioning method that uses the centroid of a cluster to represent that cluster. The centroid of a cluster is defined by the mean value of the objects inside that cluster. The quality of each cluster can be determined by the sum of squared error between its centroid and all of the objects in that cluster.

The k-means algorithm starts by placing the cluster centers at arbitrary positions and updates this position at each iteration in order to minimize the clustering error. The main drawback of this method lies in its dependency on the initial positions of the cluster centers. In regard to clustering errors, it usually obtains a local optimum solution [19]. To solve this problem, multiple runs with different initial center positions must be executed.

The k-means algorithm works as follows [12]:

1. Get the number of clusters, k,
2. Choose randomly k objects from the data set as the initial cluster centers,
3. (Re)Assign an object to a cluster which has the closest cluster center to the object,
4. Recalculate the center of each cluster,
5. Go to step 3 and repeat until there is no further change.

The requirement to specify the number of clusters, and sensitivity to noise and outliers can be seen as other disadvantages of this method.

3.1.2 Expectation Maximization (EM)

Expectation Maximization (EM) [1] is a model-based clustering algorithm. It has been shown to be an optimization method that creates an efficient statistical model of data. Unlike the k-means algorithm, EM in WEKA [11] can either automatically choose the number of clusters or still have the user determine how many clusters to create. EM tries to find clusters where the maximum likelihood of their parameters is obtained. Since it uses the maximum likelihood, there is a good chance that it may converge to a local maximum.

The algorithm is divided into two steps [12]:

- The expectation step (E-step) assigns each object to each cluster based on probabilistic clustering or fuzzy clustering.
- The maximization step (M-step) reassigns objects to clusters based on the probability distribution of each cluster.

This process runs repeatedly to minimize the clustering errors and the cluster centers converge. The disadvantage of this method is that if the probability distribution occurs too many times or data set has very few data points, then EM algorithm is very costly.

3.2 Mqr-tree Spatial Indexing

The goal of a spatial index is to optimize queries by providing efficient access to spatial objects [26]. A spatial index organizes a collection of objects (e.g., points and lines) based on their locations, and can be used to determine if a object is within a region of interest. Without using a spatial index, a sequential search (i.e., linear search) of all records in the database is needed which will increase the overall searching time.

The mqr-tree [21] is a 2-dimensional indexing method that arranges spatial objects within a 2-dimensional node based on their spatial relationship. Each object or subregion of space is represented by a minimum bounding rectangle (MBR) which is the minimum 2-dimensional bounding box or rectangle that an object or subregion of space occupies. In

comparison with other benchmarks like the R-tree [10], the mqr-tree achieves noticeable improvement in overlap, overcoverage, and number of disk accesses by introducing new techniques like node organization, insertion process, and spatial relationship rules. Overall, these improvements make the search process and data retrieval more efficient. Also, the overlap of subregions is zero when using the mqr-tree for indexing point data.

For one of our strategies, we utilize the mqr-tree region search as our searching mechanism. A mqr-tree is constructed using the cluster MBRs. A cluster MBR is the minimum bounding rectangle (MBR) of all the points in a cluster (see Section 3.3.2). Then, the mqr-tree region search is used to find the cluster MBRs that overlap with the query (see Figure 3.14).

3.2.1 Structure

Each object or subregion of space is represented by a MBR (called a node MBR) which is a set of four points (lx, ly, hx, hy). The pair (lx, ly) is the lower left coordinate and the pair (hx, hy) is the upper right coordinate of the MBR. For a data point, both the lower left coordinate and upper right coordinate are the same.

Each two-dimensional node in the mqr-tree has 5 locations as shown in Figure 3.1, including northeast (NE), northwest (NW), southeast (SE), southwest (SW), and center (CTR). Each node location accesses either:

- Leaf node: (MBR,obj_ptr) where MBR is the approximation of an object and the obj_ptr is a pointer to that object, or:
- Internal (non-leaf) node: (MBR,node_ptr) where MBR is the node MBR that encompasses all MBRs in the subtree that is referenced by the node_ptr.

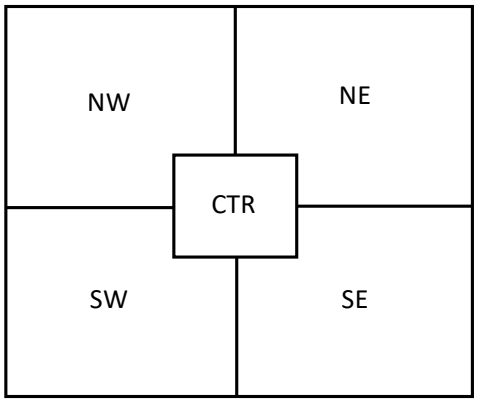

Figure 3.1: Node Layout in the mqr-tree (from [21])

The location of each object or subtree in the node is determined by the centroid of its MBR as follows:

- If an object or subtree has the same centroid as the centroid of the node MBR, then the object or the subtree is placed on the center location (CTR).
- If the centroid of an object or subtree is northeast, northwest, southeast, or southwest of the centroid of the node MBR, then the object or subtree is placed on one of the (NE, NW, SE, or SW) locations.

Figure 3.2 shows a node MBR (defined by a dashed box), which contains four objects, along with a corresponding node. Object 1 is located northwest of the node MBR centroid, while object 2 is southwest, object 3 is northeast, and object 4 is souteast of the node MBR centroid respectively.

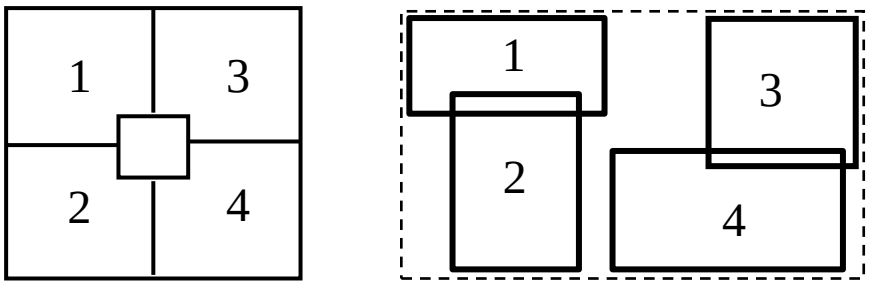

Figure 3.2: Organization of a node MBR in a two-dimensional node

3.2.2 Example

In Figure 3.3, an example of a mqr-tree for a set of point data and subtrees is provided. In the figure, the root node MBR is Mqr1 which is highlighted with bold lines. It encompasses all of the points and subtree MBRs. The Mqr1 node organizes two subtrees, Mqr2 in the NW and Mqr3 in the SE, and also a point P4 in the NE locations. Mqr2 includes all points and MBRs in the NW region. It points to p1 at the NE location, and Mqr6 at the SW location. In the same way, the Mqr3 encompasses all points and MBRs in the SE region. It points to P2 at the the NE, P3 at the NW, Mqr5 at the SE, and Mqr4 at the SW locations.

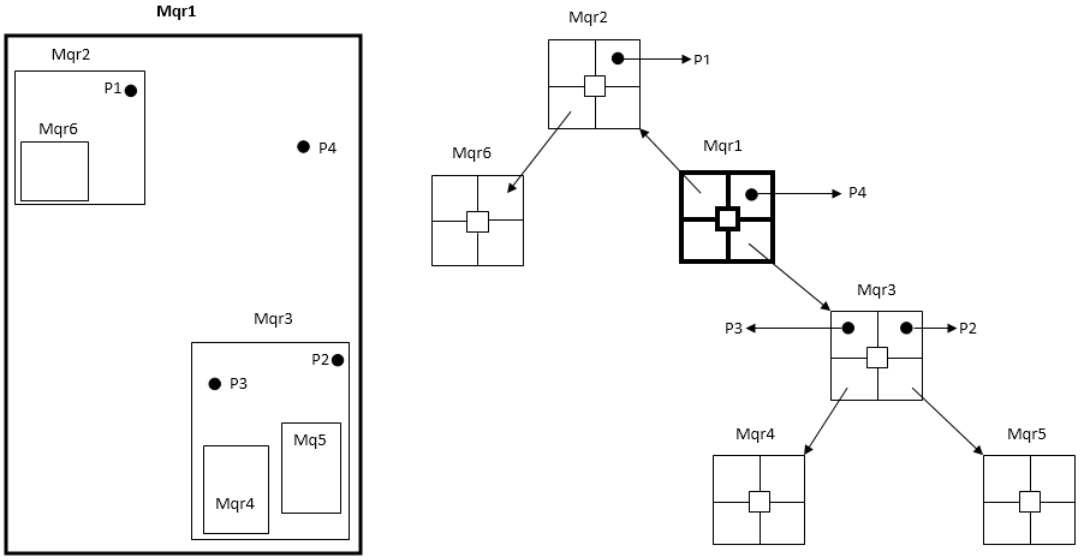

Figure 3.3: An exmaple of the mqr-tree (from [27])

3.2.3 Region search

The mqr-tree search finds all points and objects that overlap with a query region. A query region is represented as a rectangle. The search process starts from the root node by searching each of the five locations. It searches each location to find MBRs that overlap with the query region. If a location is empty then it ignores that location and continues to search on the other locations.

If the search process finds a location with a MBR that overlaps with the query then there are two possibilities:

1. If the location contains an obj_ptr, then the MBR of the associated object is returned as a result.

2. If the location contains a node_ptr, then the same search process is repeated in the relevant subtree.

3.2.4 An example of region search

In this section, we show how the mqr-tree region search works on a sample data set, shown in Figure 3.4. Each record is represented as follows:

lx : ly : hx : hy : 0 : ID

Where (lx, ly) is the lower left coordinate and (hx, hy) is the upper right coordinate of a point, 0 is elevation (which is not considered in our work), and ID is a city name or an id for an arbitrary location that is randomly generated.

```
0.11 : 68.64 : 0.11 : 68.64 : 0 : 1
0.30 : 35.99 : 0.30 : 35.99 : 0 : 2
1.37 : 47.97 : 1.37 : 47.97 : 0 : 3
1.92 : 22.64 : 1.92 : 22.64 : 0 : 4
2.40 : 32.73 : 2.40 : 32.73 : 0 : 5
3.71 : 185.36 : 3.71 : 185.36 : 0 : 6
3.77 : 139.74 : 3.77 : 139.74 : 0 : 7
4.08 : 115.77 : 4.08 : 115.77 : 0 : 8
4.26 : 20.97 : 4.26 : 20.97 : 0 : 9
4.97 : 196.34 : 4.97 : 196.34 : 0 : 10
```

Figure 3.4: A sample point data set for the mqr-tree (from[27])

In Figure 3.5, the mqr-tree is created from this sample data set.

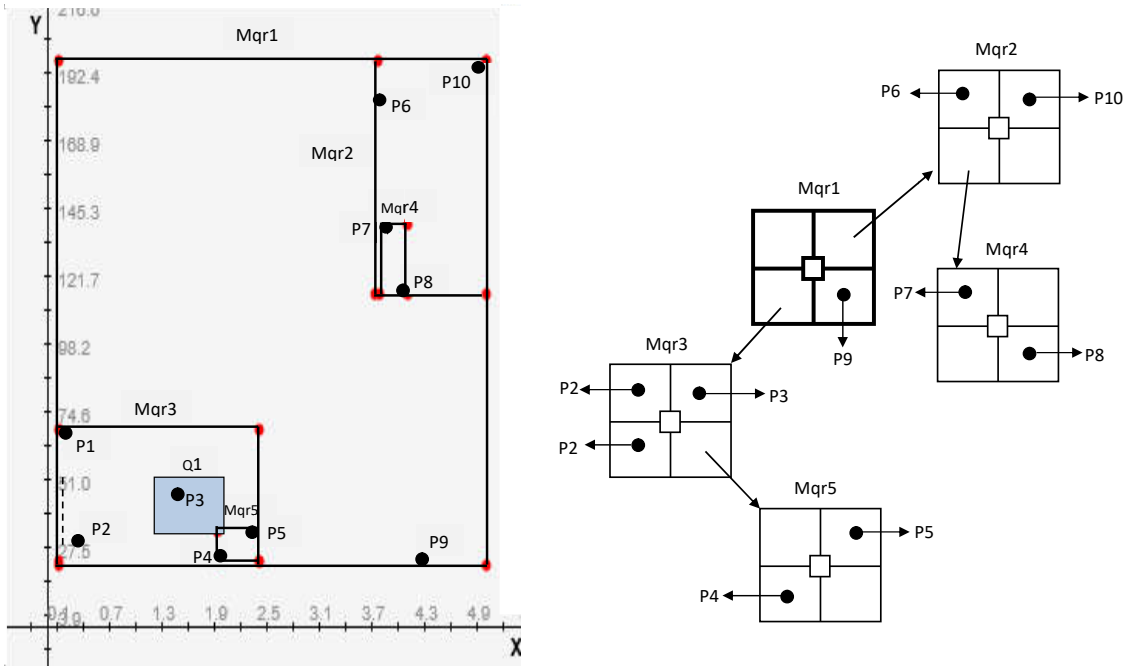

Figure 3.5: The mqr-tree for sample data in Figure 3.4 and query Q1 (from[27])

Also shown in Figure 3.5 is query Q1. The search begins from the root node Mqr1 by determining whether Q1 overlaps any of the MBRs accessible from the root node locations (NE, NW, SE, SW, CTR). As can be seen from Figure 3.5, the query Q1 overlaps Mqr3 from the SW location of Mqr1. Therefore, the search process continues in the subtree Mqr3. Within Mqr3, Q1 overlaps with NE location which contains the point P3 and the subtree Mqr5. A visit to Mqr5 results in no further overlap. Hence, the search process ends and the server returns P3 as the result of region search.

3.3 First Strategy

In this section, we introduce our first strategy. In Section 3.3.1, we provide an overall description of our first strategy. In Section 3.3.2, we define what a cluster MBR is. In Section 3.3.3, we describe how to find superMBR region. In Section 3.3.4, we describe the strategy with an example.

3.3.1 Overview

The user (i.e., client) requests data by sending queries to the server. The user sends the first query to the server. The server receives the query and utilizes a linear search to find the cluster MBRs that overlap (partially or completely) the query. After finding those MBRs, the server fetches all of the points associated with each cluster MBRs and also calculates the superMBR region that encompasses the cluster MBRs. Then, the server sends the point set and the superMBR to the client. After receiving the results, the client identifies the points that are inside the region query and displays those points to the client. For all subsequent queries, the client first compares the new query with the existing superMBR region in the following way:

- If the query is entirely contained within the superMBR, then the client updates and displays the result for the new query without the necessity of sending the query to the server.
- If the query is not entirely within the superMBR, then the client sends the query to the server to obtain new results.

The overall process of this strategy is shown in the Figure 3.6.

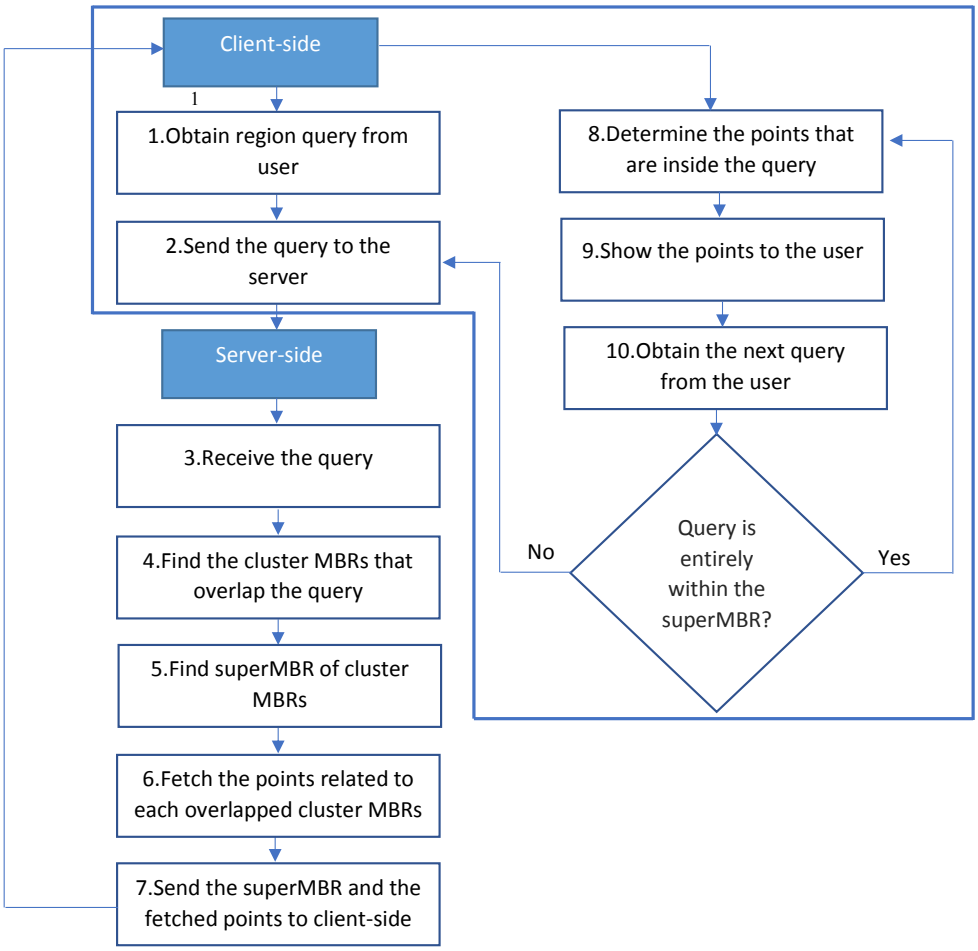

Figure 3.6: The overall process of the first strategy

[1]This step is only performed for the first query in all strategies

3.3.2 Cluster MBR

A cluster MBR is the minimum bounding rectangle (MBR) of all the points related to a cluster. Figure 3.7 shows the representation of the cluster MBRs. Figure 3.8 shows the point set for the cluster0 MBR.

Each cluster MBR is represented as:

$$lx : ly : hx : hy : 0 : \text{cluster identifier}$$

where (lx, ly) is the lower left coordinate and (hx, hy) is the upper right coordinate of the cluster MBR, 0 is elevation (which is not considered in our work), and cluster identifier is

an id for each cluster that is sorted in an ascending order.

```
118.32 : 4.83    : 141.42 : 77.61   : 0 : cluster0
55.62  : 218.54 : 86.18  : 228.24 : 0 : cluster1
1.75   : 38.69  : 56.01  : 85.18   : 0 : cluster2
166.11 : 115.14 : 204.77 : 166.11 : 0 : cluster3
212.15 : 55.62  : 228.24 : 100.89 : 0 : cluster4
60.77  : 85.50  : 112.75 : 127.78 : 0 : cluster5
100.89 : 177.06 : 154.39 : 212.15 : 0 : cluster6
4.83   : 131.63 : 51.91  : 141.42 : 0 : cluster7
```

Figure 3.7: A sample cluster MBR set on the server

```
118.32 : 77.61 : 118.32 : 77.61 : 0 : 1
123.34 : 69.35 : 123.34 : 69.35 : 0 : 2
127.79 : 60.78 : 127.79 : 60.78 : 0 : 3
131.64 : 51.91 : 131.64 : 51.91 : 0 : 4
134.87 : 42.81 : 134.87 : 42.81 : 0 : 5
137.48 : 33.51 : 137.48 : 33.51 : 0 : 6
139.45 : 24.05 : 139.45 : 24.05 : 0 : 7
140.76 : 14.48 : 140.76 : 14.48 : 0 : 8
141.42 :  4.83 : 141.42 :  4.83 : 0 : 9
```

Figure 3.8: The point set for the cluster0 MBR

3.3.3 Finding the superMBR for a query

The superMBR is the minimum bounding rectangle (MBR) that encloses all of the cluster MBRs that overlap a query. This is shown with an example. In Figure 3.9, suppose,

a client sends a query Q1 to the server.

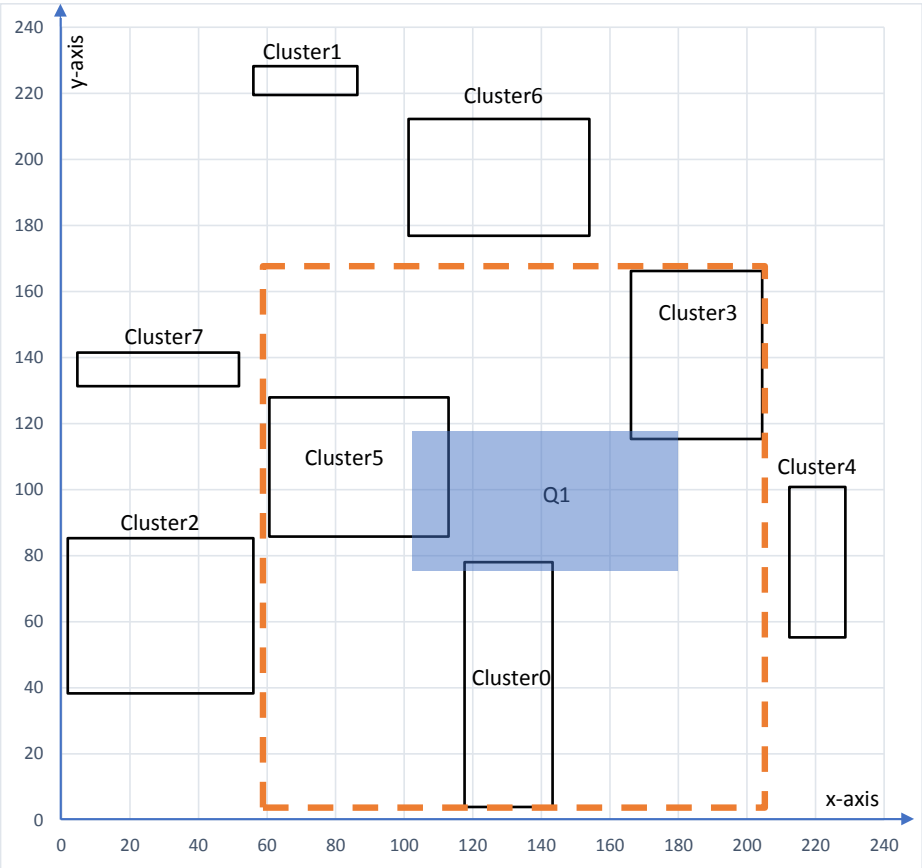

Figure 3.9: An example for defining a superMBR region for a query

The query Q1 overlaps with three cluster MBRs - cluster0, cluster3, and cluster5. The superMBR for query Q1 is the region with the dashed lines that encompasses all three cluster MBRs.

3.3.4 Example

In this section, we show an example of the execution of the first strategy for 3 consecutive queries. We use the sample cluster MBR set shown in Figure 3.7.

As shown in the Figure 3.9, after finding the superMBR region for the first query Q1 which is the minimum bounding rectangle containing the cluster0, cluster3, and cluster5 MBRs, the server fetches points associated with each cluster MBR in the superMBR and

then sends all the points and superMBR to the client. For the second query Q2 (Figure 3.10), the client first compares Q2 with the superMBR to see if it is completely inside the superMBR. As can be seen in Figure 3.10, Q2 is completely inside the superMBR region. Therefore, the client does not need to send Q2 to the server and only updates the result from the data points (i.e., the points belong to cluster3 and cluster5 which are within Q2) that are already on the client-side.

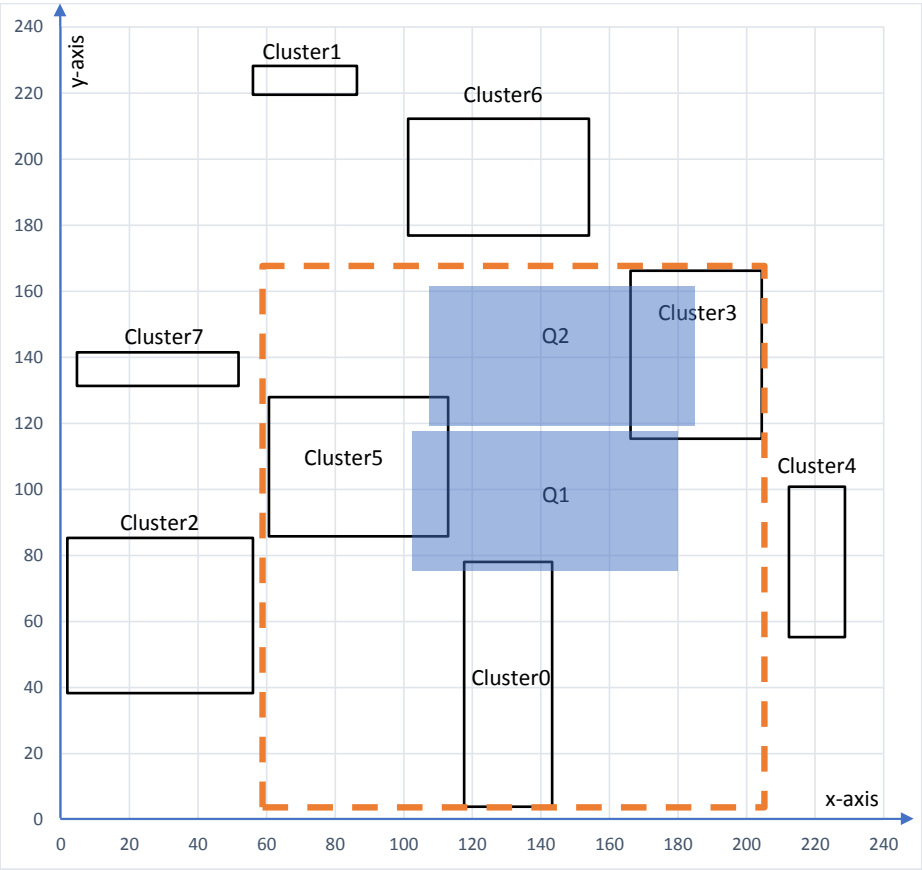

Figure 3.10: The first strategy and the second query Q2

Next, for the third query Q3 (Figure 3.11), the client compares Q3 with the superMBR before sending the Q3 to the server. This time, the superMBR does not fully encompasses Q3. As a result, the client sends Q3 to the server to obtain a new superMBR and set of data points.

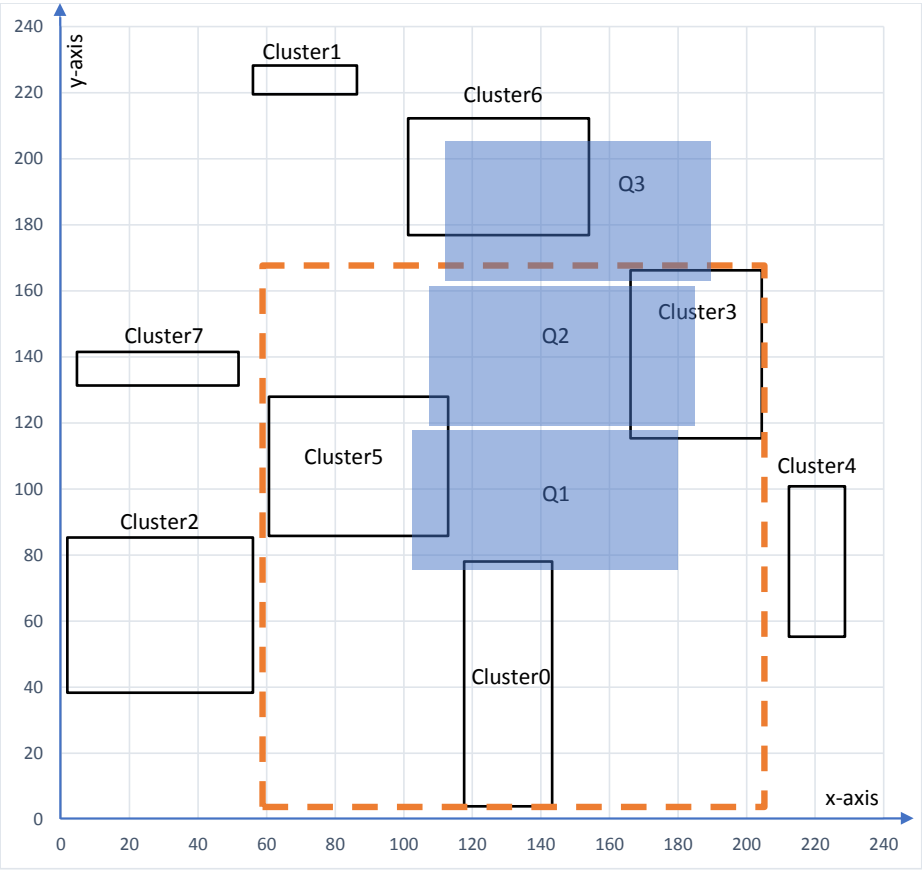

Figure 3.11: The first strategy and the third query Q3 (outside the superMBR)

The server receives Q3 and finds that the cluster3 and cluster6 MBRs overlap Q3. Therefore, the server creates a new superMBR region as shown in Figure 3.12. Then, the server fetches the points related to cluster3 and cluster6 and sends the superMBR and the fetched points to the client.

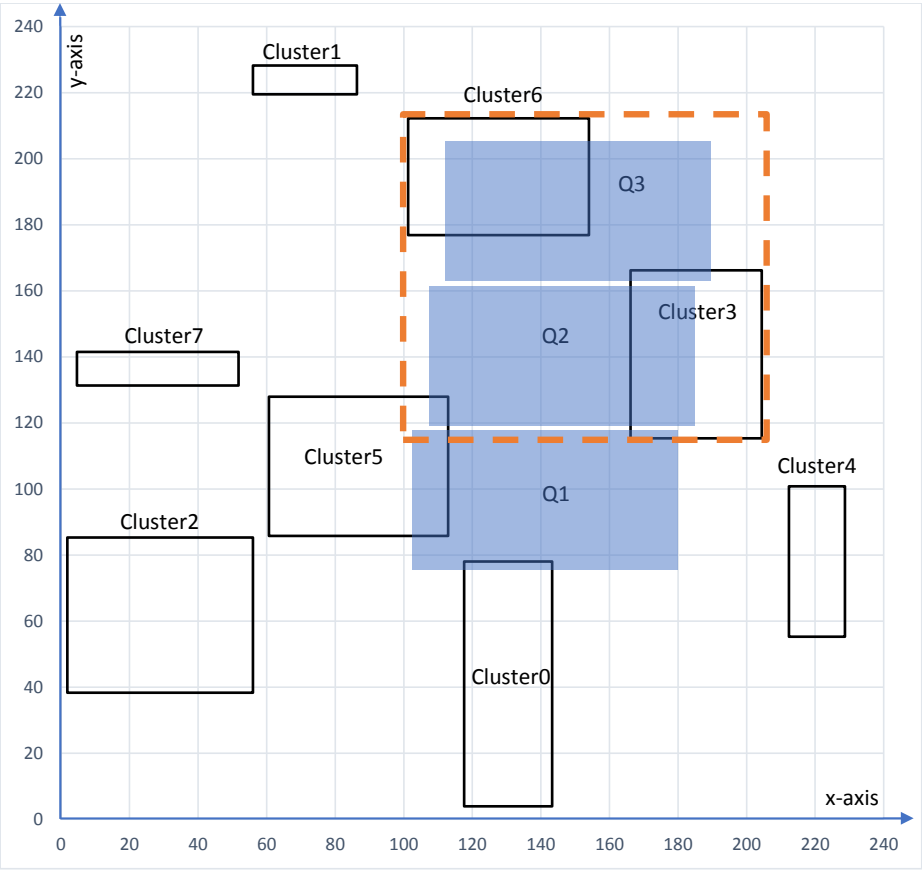

Figure 3.12: The first strategy and the third query Q3 (new superMBR)

3.4 Second Strategy

In this strategy, we use a spatial indexing structure, the mqr-tree to find overlapping cluster MBRs, instead of the linear search method on the server. There are several factors for our decision to use the mqr-tree region search in our work. First, the region search of the mqr-tree needs a lower number of node accesses over other existing spatial indexing strategies, and as a result it can reduce the query response time. Also, by utilizing the mqr-tree region search we retrieve objects based on their location directly, which can be more efficient than performing a linear search of the entire data set.

We employ the superMBR technique in this strategy as well. The rest of the process is the same as the first strategy. An overall process of this strategy is shown in Figure 3.13.

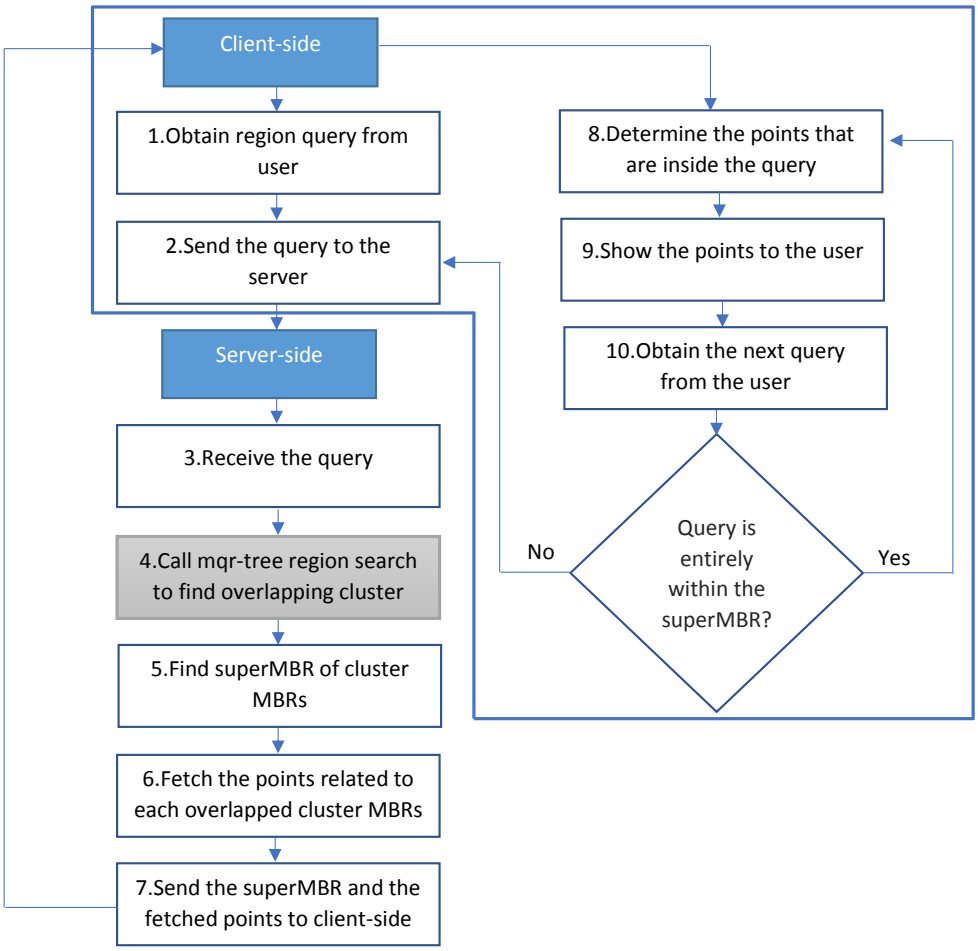

Figure 3.13: The overall process of second Strategy

3.4.1 Example

We use the same data set in Figure 3.7 to create the mqr-tree. Figure 3.14 shows the mqr-tree structure for the given data set. The root node MBR, Mqr1, is a region that encompasses the whole set of cluster MBRs. The Mqr1 node points to four MBRs, Mqr2, Mqr3, Mqr4 and Mqr5. The Mqr2 subtree is at the northeast, the Mqr3 subtree is at the southeast, the Mqr4 subtree is at the southwest, and the Mqr5 subtree is at the northwest of the root node MBR.

The client sends a query Q1 to the server. The server receives the query Q1 and uses the mqr-tree region search to perform the search process. Starting from the root node Mqr1, the mqr-tree tests to see if Q1 overlaps the NE, SE, SW, or NW locations of Mqr1 respec-

tively. At the NE location, we see that Q1 overlaps Mqr2. Therefore, the search proceeds along the subtree Mqr2, and finds that Q1 overlaps with the cluster3 MBR. Next, the search continues in the SE location of the root, Mqr3, and we see that Q1 overlaps with the cluster0 MBR. Further, the search proceeds to the SW location of the root, where Q1 overlaps with the cluster5 MBR. After that, a check of the NW location does not find any overlapping cluster MBR and consequently the search process stops. Finally, the server calculates the superMBR of the retrieved cluster MBRs (i.e., cluster3, cluster5 and cluster0 MBRs), fetches the points inside the superMBR, and sends all the points and superMBR to the client. For the subsequent queries, the client first checks the query with the superMBR and it issues a new query to the server only when the query is not within the superMBR.

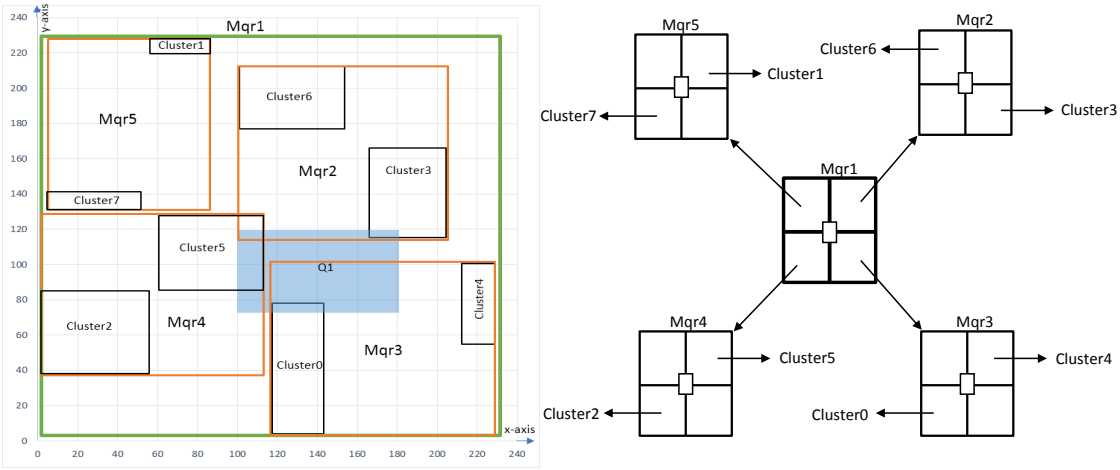

Figure 3.14: The mqr-tree for the sample cluster MBRs

3.5 Baseline Strategy

This strategy is provided for comparison with other two strategies for evaluation purposes. In this strategy, a client sends all of the queries to the server without any comparison between the superMBR and the queries on the client-side. We use this strategy for comparison against other two strategies. The overall process of this strategy is shown in Figure 3.15.

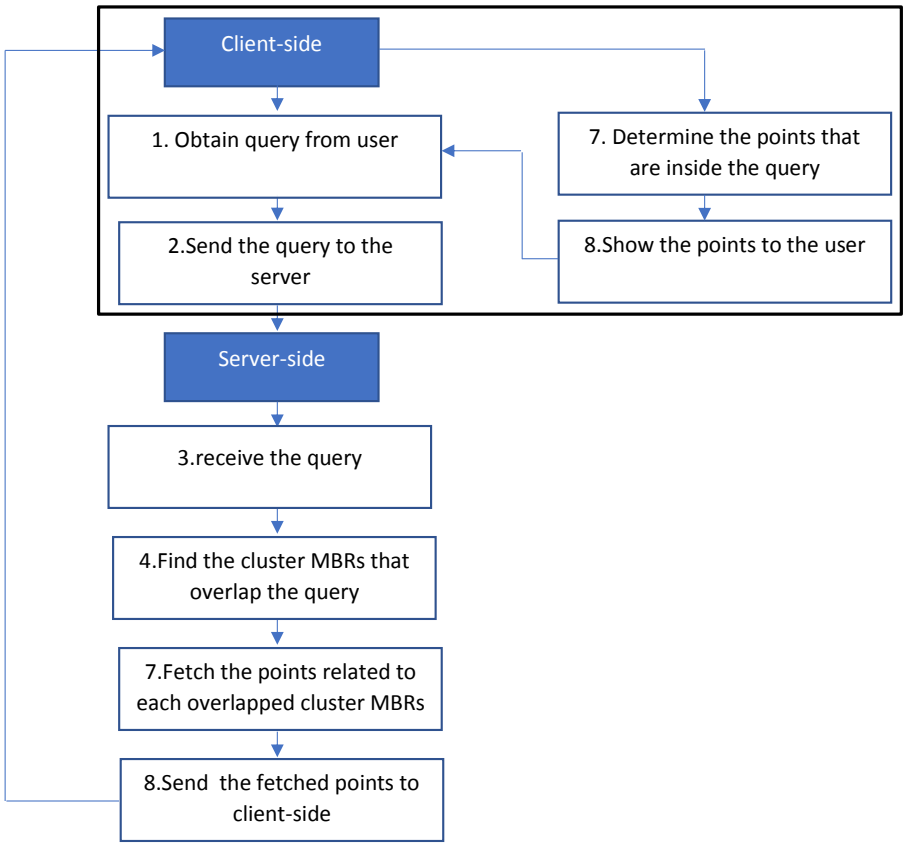

Figure 3.15: The overall process of baseline strategy

This strategy has more data transmission cost and server workload than the other two approaches.

3.5.1 Conclusion

In this chapter, we present our three continuous region query processing strategies over clustered data set. We discuss performance evaluation of these strategies in the next chapter.

Chapter 4

Experiments and Results

In this chapter, we assess the performance of our spatial query processing strategies by performing several tests. The result shows that our first and second strategies reduce the data transmission cost and running time of our application over the baseline strategy. The data transmission cost and the number of queries that need to be sent to the server are the same for the first and the second strategies. The running time of the first strategy is better than the second strategy for smaller data sets and in the larger data sets, the second strategy outperforms the first strategy.

The rest of this chapter is organized as follows. In Section 4.1, we illustrate the settings of our experiments. In Section 4.2, we describe the data sets we used for our experiments. In Section 4.3, we introduce the measures that we used to evaluate the performance of our strategies. In Sections 4.4, 4.5, 4.6, and 4.7 we present the performance results of our strategies for various tests. In Section 4.8, we discuss some preliminary results for our New Zeland and Waikato data sets, and the reasons why we changed our data sets.

4.1 Experiment Setup

We simulated a general client-server application for our experiment. In our application, a moving client sends queries to a server. The server receives the queries, processes them, and sends the results back to the client. In the client-side, each time the client wants to send a new query (except the first query) to the server it first checks whether the query is completely inside the superMBR region or not. If the superMBR fully encompasses

the query then the client updates the data points without sending the query to the server. Otherwise, it sends the query to the server to obtain a new result.

We evaluated the performance of our strategies based on different tests:

- Varying number of points and density
- Varying query sizes
- Different clustering algorithms
- Different number of clusters for a data set

We ran each test ten times and calculated the average value of the results. We used the C# programming language and the ASP.NET MVC framework for implementing the server side and for client side we utilized JavaScript and jQuery. All the experiments were performed on an Intel(R) Core(TM) i7-4510U CPU @2.00 GHz 2.60 GHz with 8 GB RAM computer running the Microsoft Windows 8.1 operating system.

4.2 Data sets

We use twelve synthetic data sets in our experiment. The number of points in the data sets are 500, 1000, 5000, 10000, 50000, and 100000 respectively. The points are randomly generated by using uniform distribution and exponential distribution. The format of each point in the data set is as follows:

$$lx : ly : hx : hy : 0 : ID$$

Where (lx, ly) is the lower left coordinate and (hx, hy) is the upper right coordinate of a point (the lower coordinate is the same as the upper coordinate for each point), 0 is elevation (we did not consider this factor in our work), ID is a city name or an id for an arbitrary location that we randomly generated for our purposes.

For a data file with N points, the space that these points occupied is ($\sqrt{N}$ * 10 x $\sqrt{N}$ * 10). Figure 4.1 shows examples of how 25 points can be uniformly and exponentially distributed over an (50 x 50) area.

Also, for each data set with N points, approximately $\sqrt{N}$ points are created that traverse along the diagonal path, which will serve as a query set. In order to form a region around a point (i.e., conduct a region query), we add and subtract an offset to the x and y coordinates of the point. Overall, we create six query sets in the same format as the data sets.

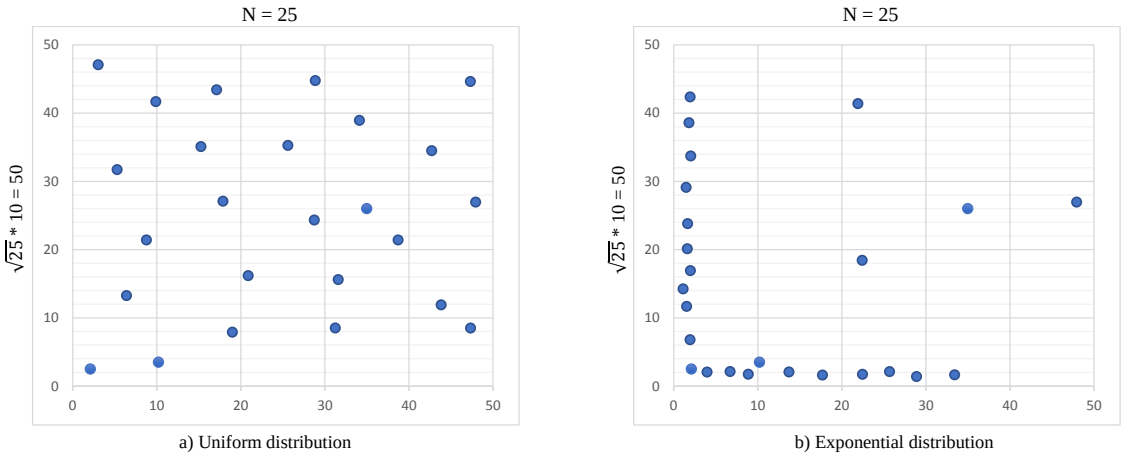

Figure 4.1: Example of 25 points uniformly and exponentially distributed over an (50 x 50) area

We selected the k-means and EM (Expectation Maximization) algorithms to cluster our data sets. The k-means is a centroid-based clustering algorithm which is popular because of its simpilicity and the ability to specify the desired number of clusters. EM is an efficient optimization algorithm which maximize the likelihood for each cluster's parameters. In addition, we use the WEKA 3.7 data mining software [11] to perform the clustering task. It is a well-established machine learning software which contains a stable implementation of the k-mean and EM algorithms. After clustering each data set, we create and store the cluster MBRs for each data set in a file using the same format as the data sets. Also, we save the points associated with each cluster in a separate spreadsheet in an Excel file in order to fetch the points from the file whenever a query overlaps with a cluster MBR. The

Excel format is used due to Excel processing libraries available in C#.

4.3 Performance Measures

We evaluate our query processing strategies based on the following performance metrics:

1. Number of queries that must be sent to the server: This is the overall number of queries that a client sends to the server. Before issuing any query (except the first query), the client checks the query with the superMBR region. If the query is completely inside the superMBR then there is no need to send the query. This measure always has the same value for our first and second strategies.

2. Running time: This time is calculated in millisecond from when the server receives the query, identifies the results and sends them to the client until the client determines the points inside the query. Since the client and server are on the same machine (i.e., local hosting), the network latency is negligible. Figure 4.2 shows the calculation of this time.

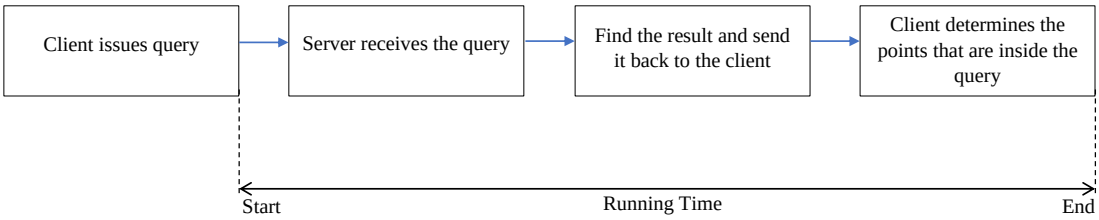

Figure 4.2: The running time metric

3. Data transmission cost: This is the cost of sending the result from the server to the client for each query. This cost is calculated as follows:

   – 8 bytes per floating number

   – 4 bytes per integer number

   – 1 byte per character

4. Number of clusters fetched from the server: This is the total number of overlapping cluster MBRs that the server finds and sends to the client as a part of query result.

4.4 Case 1: Varying number of points (and density)

For this experiment, we evaluate the strategies by varying the number of points and density. We use all twelve synthetic data sets mentioned earlier. For constants, 1) we chose the k-mean clustering algorithm to cluster each data set to 10 clusters, and 2) we use 10% of the data space for the query size. The reason we selected the k-means algorithm is the simplicity of k-means to apply to large data sets, and also it is always guaranteed to converge to a result. The ideal choice of k (i.e., number of clusters) is often difficult and opaque, however we decided to cluster our data set to 10 clusters in order to maximize the efficiency of clusters and minimize the clustering errors. We use 10% of the data space area for the query size in order to subsequently have a safe region (i.e., the superMBR) that is well-balanced (i.e., not too small and not too big). If the safe region is too small then the client usually needs to send more queries to the server, which will increase the data communication cost. On the other hand, if the safe region is too big then more non-result data points will be sent to the client and consequently it will increase the computational cost on the client-side. Also, there may not be enough storage available on the client for the result.

4.4.1 The number of queries sent to the server

Table 4.1 displays the number of queries sent to the server for this experiment. We can see in Table 4.1 that the number of queries that are sent to the server by the first and second strategies is far fewer than our baseline strategy. It is specifically noticeable for all uniform data sets and bigger exponential data sets. The reason is that, in the first and second strategies our proposed safe region method (i.e., the superMBR) enables the client to move inside this region for an extended time without the need to issue any queries to the server.

The superMBR region is formed from the minimum bounding rectangle of cluster(s) which overlapped with the query and not from the point set directly. This can lead to larger regions for the query to move around on the client. As a result, this will significantly reduce the number of queries that needs to be sent to the server. By sending fewer queries to server, we reduce the server workload and also data transmission cost.

Table 4.1: The number of queries sent to the server in Case 1

Data sets	Queries sent to the server	
	Strategy1&2	Baseline
up_500	5	19
up_1000	3	28
up_5000	4	63
up_10000	4	89
up_50000	3	198
up_100000	4	283
ep_500	13	19
ep_1000	12	28
ep_5000	6	63
ep_10000	4	89
ep_50000	4	198
ep_100000	4	283

4.4.2 The running time

In Table 4.2, the running time of our strategies for different data set sizes is presented. Overall, we can see from the figure that as the number of points in the data sets grow the running time of the strategies increase as well. In addition, our first and second strategies outperform our baseline strategy in most cases, except the ep_500 and ep_1000 data sets, where the baseline strategy has better performance than our second strategy. The reason is that the number of queries that we sent to the server is higher in the ep_500 and ep_1000 data sets than the other data sets (see Table 4.1). Also, the baseline strategy does not perform a superMBR comparison, which makes the baseline strategy faster for those two small data sets. Furthermore, our second strategy, which uses the mqr-tree region search on the server

to find the overlapping clusters, has a lower running time than the first strategy for all of the 50000 and 100000 points data sets, but not for the smaller data sets. One possible reason for this is the extra time required to construct the mqr-tree before we can use it for region searches. It is noticeable in smaller data sets, but for the bigger data sets, the query response time is not affected as much, since the mqr-tree is only built once but searched more often.

Table 4.2: The running time of proposed strategies for our data sets [3]

Data sets	Running Time (ms)		
	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	3358.6	34732.6
ep_100000	9101.8	9023.1	136944.8

4.4.3 Data transmission cost

In this section, we compare the data transmission cost of our strategies over different data sets. As can be seen from Table 4.3, the first and second strategies have a far lower data transmission cost than our baseline strategy. This happens because in the first and second strategies, the user sends a lower number of queries to the server (as shown in Table 4.1), and consequently fewer query results are required to be sent back to the user. This

[3] Strategy 1 is the approach using linear search, and strategy 2 is the approach using the mqrtree

subsequently results in a significantly lower amount of data being transmitted.

Table 4.3: The data transmission cost for various data sets

Data sets	Data transmission cost	
	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

4.5 Case 2: Varying query sizes

In this test, we try different query sizes including 2.5%, 5%, 7.5%, and 10% of the data space over the up_10000 data set. We chose the up_10000 data set in order to have a sufficient number of points overall and in our clusters, but at the same time achieve lower response times (see Figure 4.2). As before, we clustered the data set into 10 clusters using the k-means algorithm.

4.5.1 The number of queries sent to the server

Table 4.4 shows the number of queries that has been sent to the server for different query sizes. For the 2.5%, 7.5%, and 10% query sizes, the first and second strategies only send four queries and for 5% query size five queries are sent to the server. All are significantly lower than the baseline strategy as we send the results of all queries in the baseline strategy.

The reason for sending five queries in 5% query size is that the superMBR which is created for the 5% query size is smaller than the other query sizes. This is because, the cluster MBRs which are fetched for the 5% are smaller or closer to each other, and subsequently this leads to a smaller superMBR. Smaller superMBRs will often lead to a higher number of queries sent to the server. We can see this in the 5% query sizes, however the 2.5%, 7.5%, and 10% are not effected as much in this case.

Table 4.4: The number of queries sent to the server in Case 2

Query sizes	Queries sent to the server	
	Strategy1&2	Baseline
2.5%	4	89
5%	5	89
7.5%	4	89
10%	4	89

4.5.2 The running time

The running time of our strategies for this case is shown in Figure 4.3. As can be seen from the figure, the first and second strategies have better performance than the baseline strategy for all query sizes. The first strategy has better performance than the second strategy for all query sizes. As we discussed in Section 4.4.2, the slight increase in running time of our second strategy in comparison with our first strategy is because of the necessity to construct the mqr-tree before we can use its region search which adds up this extra time. We can also see that by sending fewer queries to the server (see Table 4.4) the query response time is reduced as well.

Query size	Running Time (ms)		
	Strategy1	Strategy2	Baseline
2.5%	161.3	251.3	1660.5
5%	259.2	321.2	1997.2
7.5%	377.5	411.0	2533.7
10%	526.7	556.5	3024.0

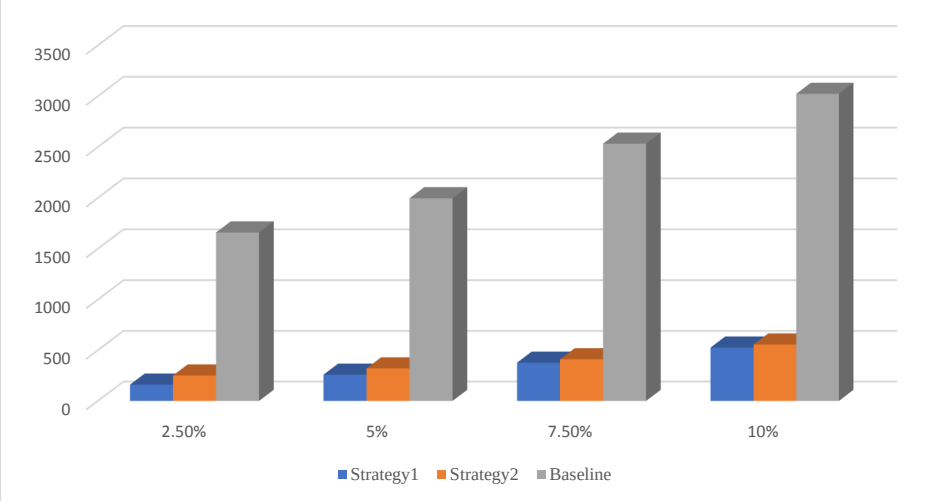

Figure 4.3: The running time of the strategies for different query sizes

4.5.3 Data transmission cost

Figure 4.4 shows the data transmission cost of our strategies for different query sizes. We can see that in the first and second strategies the data transmission cost for 5% query size is larger than 7.5% and 10% query sizes and smaller than 2.5% query size. This is due to the number of queries sent to the server. As can be seen from Table 4.4, for the 5% query size, five queries are sent to the server (the highest number among all query sizes) and therefore more data is being sent to the client in this case. Also, the data transmission cost for the 7.5% and 10% query sizes is the same, which means that the same superMBRs are created for these cases.

Query size	Data Transmission Cost	
	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

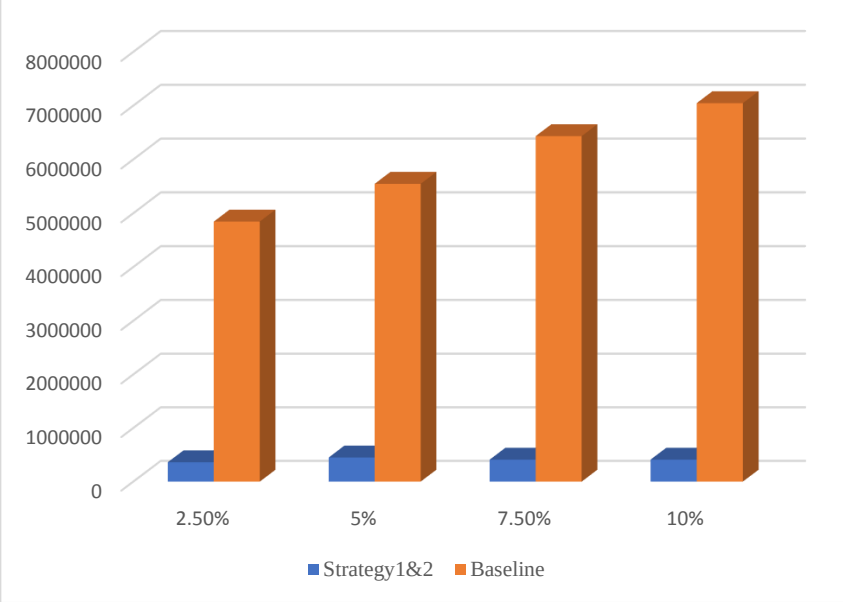

Figure 4.4: The data transmission cost for various query sizes

4.6 Case 3 : varying clustering algorithm

For this experiment, we compared the use of the k-means and EM clustering algorithms in our strategies. As before, we chose the up_10000 data set and clustered it into 10 clusters (by both the k-means and EM). Also, we clustered the data set by letting the EM algorithm in WEKA automatically select the number of clusters by cross validation, which is eight clusters. Finally, as before, we used a query size of 10% of the data space for this experiment.

4.6.1 The number of queries sent to the server

In Table 4.5, we show the number of queries which are sent to the server for different clustering algorithms. The number of queries sent to the server in the first and second strategies for the k-means and EM algorithms (number of clusters set to 10) is 4. For the EM (number of clusters selected by WEKA) this number is 3 which shows EM leads to better performance than both the k-means and EM (10) algorithms. One possible reason is the fact that EM is an optimization algorithm and in combination with WEKA's cross validation, it converges to a better result than the k-means algorithm or EM (10). Also, the selection of fewer clusters by WEKA can lead to larger superMBRs because as we mention before, the superMBR is formed from the cluster MBRs that overlapped with the query not from the point set directly. Larger superMBRs can lead to a fewer number of queries that need to be sent to the server.

Table 4.5: The number of queries sent to the server in Case 3

Clustering algorithms	Queries sent to the server	
	Strategy1&2	Baseline
k-mean (10 clusters)	4	89
EM (Default)	3	89
EM (10 clusters)	4	89

4.6.2 The running time

The running time of our strategies for varying clustering algorithms is shown in Figure 4.5. The first strategy has a slightly better performance than the second strategy in all clustering algorithms because of additional time needed to construct the mqr-tree in the second strategy. Both first and second strategies outperform the baseline strategy. The reason is that EM (default) leads to the lowest running time in our algorithms since only three queries are sent to the server in comparison to other algorithms which is four (see Table 4.5). As a result, this also reduces the running time of the strategies.

Clustering algorithms	Running Time (ms)		
	Strategy1	Strategy2	Baseline
k-mean	529.8	548.3	3028.4
EM (Default)	302.8	322.8	3689.0
EM (10 clusters)	591.6	621.1	3752.1

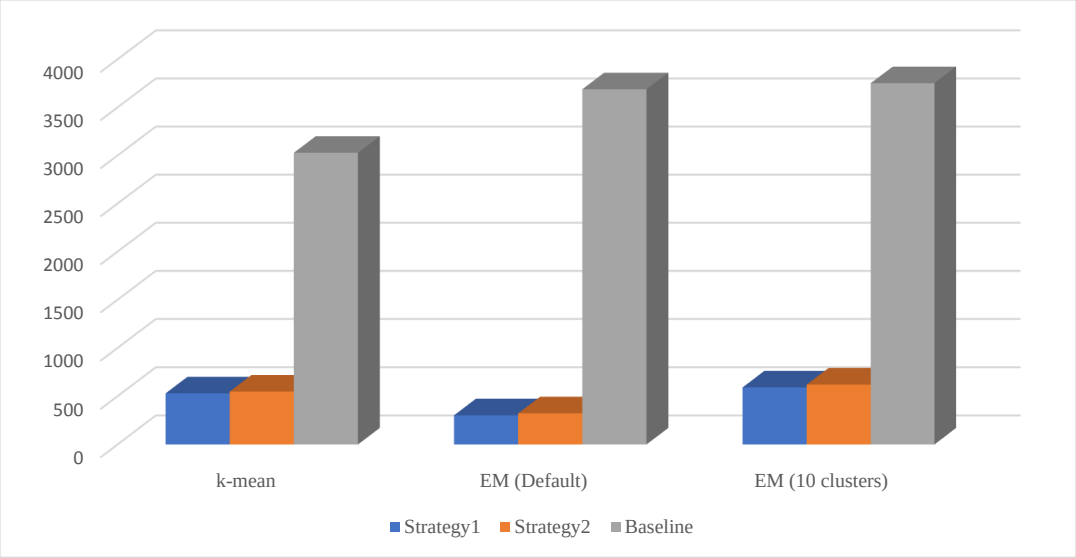

Figure 4.5: The running time for different clustering algorithm

4.6.3 Data transmission cost

Figure 4.6 shows the data transmission cost of our strategies over the up_10000 data set when clustered by the k-means and EM clustering algorithms. As can be seen from the figure, the EM (default) has the lowest data transmission cost (382997 bytes), then the k-mean algorithm (410451 bytes) and finally EM (number of clusters set to 10) with 521994 bytes. Again, the reason is related to the number of queries that sent to the server (see Table 4.5). Also, we can see that EM with the option of 10 clusters is not clustering the data set efficiently as it has the highest data communication cost among all the strategies. As we mentioned earlier, the higher number of clusters can lead to smaller superMBRs. Smaller superMBRs will often increase the number of queries which is needed to send to the server and consequently this will increase the data transmission cost.

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

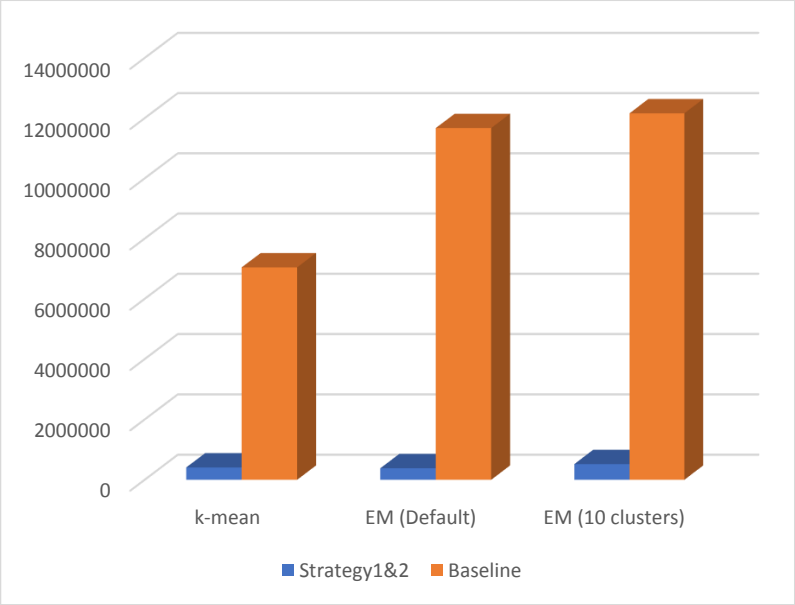

Figure 4.6: The data transmission cost for a data set clustered by varying clustering algorithms

4.7 Case 4 : varying the number of clusters

In this test, we cluster the up_10000 data set to different number of clusters 10, 20, ..., up to 100. As before, the queries occupy 10% of the data space, and the k-means clustering is used.

4.7.1 The number of queries sent to the server

Table 4.6 shows the number of queries sent to the server for different numbers of clusters. As we can see from the table, the number of queries sent to the server slightly increases when the number of clusters increases. However, it is still significantly lower than when we

compare the result with the baseline strategy. Again, the reason is that as we increase the number of clusters the superMBR will be smaller and therefore more queries needed to be sent to the server due to the query moving out of the superMBR region sooner. Since the number of points is constant (i.e., the up_10000 data set), the higher number of clusters the smaller the superMBR will be.

Table 4.6: The number of queries sent to the server in Case 4

Number of clusters	Queries sent to the server	
	Strategy1&2	Baseline
10	4	89
20	5	89
30	9	89
40	8	89
50	8	89
60	10	89
70	10	89
80	10	89
90	12	89
100	11	89

4.7.2 The running time

In Figure 4.7, we can see the running time of our strategies for different numbers of clusters. The result shows that the first strategy has a better query response time for all the data sets compared with the second strategy. As mentioned previously, this is because of the extra time required to construct the mqr-tree before we can use it for region searches. Also, we can see that the first and second strategies outperform the baseline strategy due to sending fewer number of queries to the server. In addition, the running times of the first and second strategies fluctuates as the number of clusters grows. On the other hand, the running time of the baseline strategy decreases as the number of cluster grows. One possible reason is that for the baseline strategy, as the number of clusters increase, the superMBRs are smaller with fewer points and subsequently it takes less time to send the data to the client.

For the first and second strategies since not all queries need a new superMBR, the ones being requested vary. Hence, we can see that there is fluctuations in the running time of the first and second strategies as the number of clusters grows.

Number of clusters	Running Time (ms)		
	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

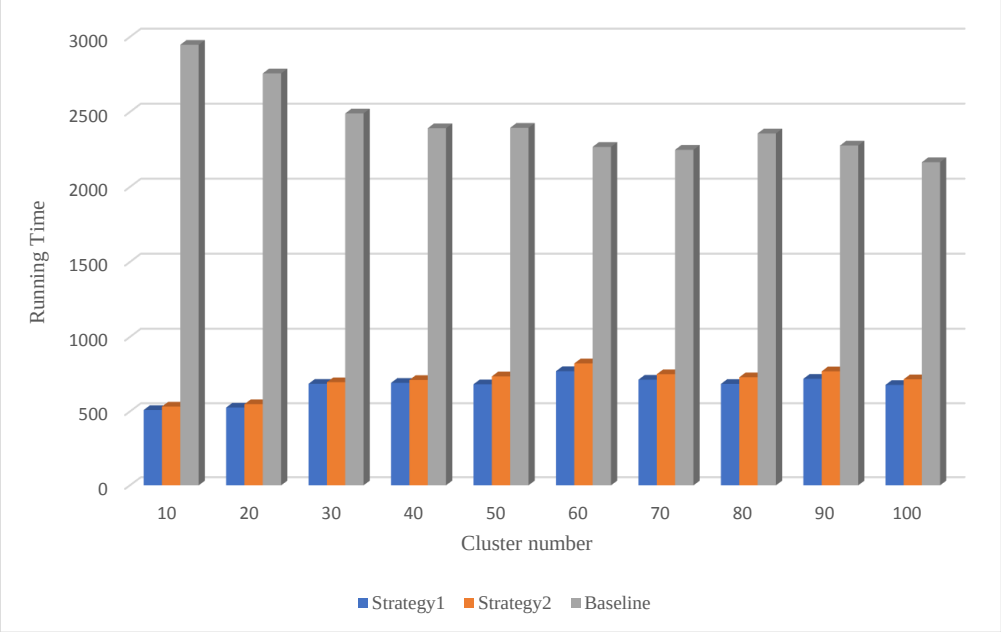

Figure 4.7: The running time for varying number of clusters

4.7.3 Data transmission cost

Figure 4.8 illustrates the data transmission cost of our strategies for different number of clusters. As we can see from the figure, the data transmission cost decreases as the number of clusters grows. It shows that the smaller clusters lead to less data transmission overall. One reason for this can be that by increasing the number of clusters the superMBR region will be smaller and this consequently will decrease the data transmission between the server and clients. Also, we can see that in the baseline strategy the data transmission cost uniformly decreases as the number of clusters grow. However, in the first and second strategies there are fluctuations. Again, one possibility would be the different size of the superMBR region for each case, and also the number of queries sent to the server.

Number of clusters	Data transmission cost	
	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

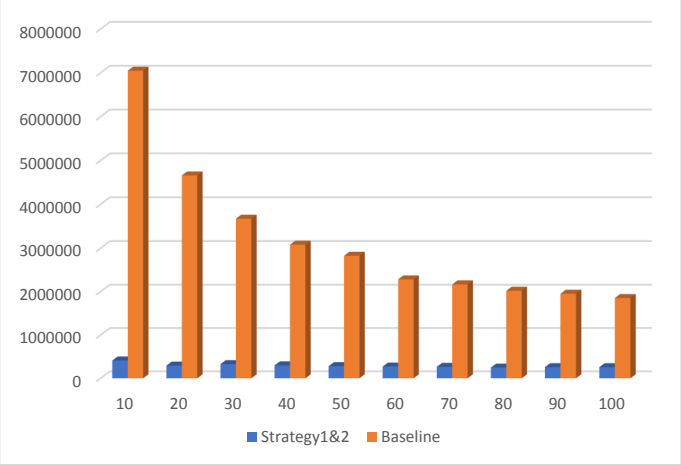

Figure 4.8: The data transmission cost for different number of clusters

4.8 New Zealand North Island and Waikato data sets

Initially, we performed different experiments with several real data sets which are drawn from two regions of New Zealand (North Island and Waikato), and we used query points that are issued and collected from some paths in those regions. As the query points are too close to each other, the superMBR that we created for our clustered data sets was too big, and therefore, it significantly reduced the number of queries (to only one) that were needed to send to the server. As a result of issuing only one query to the server for all the data sets, we decided to change our data sets to twelve different synthetic data sets and corresponding query sets that we generated with uniform and exponential distributions.

4.9 Conclusion

We proposed two moving query processing strategies over clustered data sets in Chapter 3. In this Chapter, we present the evaluation results of our strategies for different scenarios. Our first and second strategies significantly reduce the data transmission cost and the query response time of our application over the baseline strategy. Also, the second strategy has better performance in data transmission cost for bigger data sets compared with the first strategy.

Chapter 5

Conclusion

We propose two query processing strategies for location-based services. Both strategies are for a moving region query over static data sets which are clustered. In a moving query processing environment, users/clients continuously issue queries to a server, and the server processes the queries and returns the results to the users. In our first strategy, the server uses a linear search function to find the cluster MBRs that overlap with the queries. After finding the cluster MBRs, the server calculates a safe region (i.e., superMBR) and sends it along with the points data set to the user. In the client-side each time the user wants to issue a new query, first, it checks the query with the superMBR region. If the query resides in the superMBR, then it updates the query result without sending the query to the server. Otherwise, a new query will be issued to the server. In our second strategy, we replace the linear search method with the region search of a 2-dimensional spatial indexing structure, the mqr-tree, to attempt to speed up the overall process. Finally, we give a baseline strategy, where we dismiss the safe region method, and therefore all the queries will be directly sent to the server (linear search method is used on the server to find the overlapping cluster MBRs).

We asses our strategies with various performance testing. We found noticeable improvements in server workload (i.e., the number of queries sent to the server), data communication cost, and query response time in our first and second strategies over our 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 response time over our first strategy in bigger data sets.

5.1 Future work

In the future, we can work on the moving region queries over the moving data points instead of static data points since this type of queries has many applications in location-based services. In addition, we can expand our work into k nearest neighbor (kNN) queries. Massive work have been done in kNN queries, as it is one of the most popular types of continuous query processing. We will also try to protect the privacy of our queries (i.e., location privacy) by applying some cloaking algorithms.

Bibliography

[1] Adebisi Adigun, Elijah Omidiora, and Stephen Olabiyisi. An exploratory study of k-means and expectation maximization algorithms. *British Journal of Mathematics and Computer Science*, 2:62–71, jan 2012.

[2] Haidar Al-Khalidi, David Taniar, John Betts, and Sultan Alamri. Efficient monitoring of moving mobile device range queries using dynamic safe regions. In *Proceedings of International Conference on Advances in Mobile Computing & Multimedia*, MoMM '13, pages 351:351–351:360, New York, NY, USA, 2013. ACM.

[3] Haidar AL-Khalidi, David Taniar, John Betts, and Sultan Alamri. On finding safe regions for moving range queries. *Mathematical and Computer Modelling*, 58(56):1449 – 1458, 2013.

[4] Jie Bao, Chi-Yin Chow, Mohamed F. Mokbel, and Wei-Shinn Ku. Efficient evaluation of k-range nearest neighbor queries in road networks. In *Proceedings of the 2010 Eleventh International Conference on Mobile Data Management*, MDM '10, pages 115–124, Washington, DC, USA, 2010. IEEE Computer Society.

[5] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangles. *SIGMOD Rec.*, 19(2):322–331, may 1990.

[6] Hyung-Ju Cho, Rize Jin, and Tae-Sun Chung. A collaborative approach to moving k-nearest neighbor queries in directed and dynamic road networks. *Pervasive and Mobile Computing*, 17, Part A:139 – 156, 2015.

[7] Chi.Yin Chow, Mohamed F. Mokbel, and Walid G. Aref. Casper, query processing for location services without compromising privacy. *ACM Trans. Database Syst.*, 34(4):24:1–24:48, dec 2009.

[8] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, jan 2008.

[9] Chenglin Fan, Jun Luo, Wencheng Wang, and Binhai Zhu. Voronoi diagram with visual restriction. *Theoretical Computer Science*, 532:31 – 39, 2014. Frontiers in Algorithmics.

[10] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. *SIGMOD Rec.*, 14(2):47–57, jun 1984.

[11] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, nov 2009.

[12] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[13] Tanzima Hashem, Lars Kulik, and Rui Zhang. Countering overlapping rectangle privacy attack for moving knn queries. *Information Systems*, 38(3):430 – 453, 2013.

[14] Jiun-Long Huang and Chen-Che Huang. A proxy-based approach to continuous location-based spatial queries in mobile environments. *IEEE Transactions on Knowledge and Data Engineering*, 25(2):260–273, Feb 2013.

[15] Sergio Ilarri, Eduardo Mena, and Arantza Illarramendi. Location-dependent query processing: Where we are and where we are heading. *ACM Comput. Surv.*, 42(3):12:1–12:73, mar 2010.

[16] Changqing Ji, Zhiyang Li, Wenyu Qu, Yujie Xu, and Yuanyuan Li. Scalable nearest neighbor query processing based on inverted grid index. *Journal of Network and Computer Applications*, 44:172 – 182, 2014.

[17] Hyeong-il Kim and Jae-woo Chang. k-nearest neighbor query processing algorithms for a query region in road networks. *Journal of Computer Science and Technology*, 28(4):585–596, 07 2013.

[18] Chuanwen Li, Yu Gu, Jianzhong Qi, Ge Yu, Rui Zhang, and Wang Yi. Processing moving knn queries using influential neighbor sets. *Proc. VLDB Endow.*, 8(2):113–124, oct 2014.

[19] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451 – 461, 2003. Biometrics.

[20] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 218–238, New York, NY, 1990. Springer New York.

[21] Marc Moreau and Wendy Osborn. mqr-tree: A 2-dimensional spatial access method. *arXiv preprint*, arXiv/1212.1469, 2012.

[22] Kwangjin Park, Moonbae Song, and Chong-Sun Hwang. Location-based services for dynamic range queries. *Journal of Communications and Networks*, 7(4):478–488, Dec 2005.

[23] Zhou Shao and David Taniar. Enhanced range search with objects outside query range. *World Wide Web*, 18(6):1631–1653, 2015.

[24] Zhou Shao, David Taniar, and Kiki Maulana Adhinugraha. Range-knn queries with privacy protection in a mobile environment. *Pervasive and Mobile Computing*, 24:30 – 49, 2015. Special Issue on Secure Ubiquitous Computing.

[25] Mehdi Sharifzadeh and Cyrus Shahabi. Vor-tree: R-trees with voronoi diagrams for efficient processing of spatial nearest neighbor queries. *Proc. VLDB Endow.*, 3(1-2):1231–1242, sep 2010.

[26] Shashi Shekkar and Sanjay Chawla. *Spatial databases: a tour*. Prentice Hall, Upper Saddle River, NJ, 2003.

[27] Shauli Sarmin Sumi. Continuous spatial query processing in mobile information systems, 2017.

[28] David Taniar and Wenny Rahayu. A taxonomy for nearest neighbour queries in spatial databases. *Journal of Computer and System Sciences*, 79(7):1017 – 1039, 2013.

[29] David Taniar and Wenny Rahayu. A taxonomy for region queries in spatial databases. *Journal of Computer and System Sciences*, 81(8):1508 – 1531, 2015.

[30] Kun-Lung Wu, Shyh-Kwei Chen, and Philip S. Yu. Efficient processing of continual range queries for location-aware mobile services. *Information Systems Frontiers*, 7(4):435–448.

[31] Jianliang Xu, Xueyan Tang, Haibo Hu, and Jing Du. Privacy-conscious location-based queries in mobile environments. *IEEE Transactions on Parallel and Distributed Systems*, 21(3):313–326, March 2010.

[32] Man Lung Yiu, Eric Lo, and Duncan Yung. Authentication of moving knn queries. In *2011 IEEE 27th International Conference on Data Engineering*, pages 565–576, April 2011.

[33] Duncan Yung, Yu Li, Eric Lo, and Man Lung Yiu. Efficient authentication of continuously moving k nn queries. *IEEE Transactions on Mobile Computing*, 14(9):1806–1819, Sept 2015.

[34] Duncan Yung, Eric Lo, and Man Lung Yiu. Authentication of moving range queries. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM 12, pages 1372–1381, New York, NY, USA, 2012. ACM.

[35] Duncan Yung, Man Lung Yiu, and Eric Lo. A safe-exit approach for efficient network-based moving range queries. *Data and Knowledge Engineering*, 72:126 – 147, 2012.

[36] Jun Zhang, Manli Zhu, Dimitris Papadias, Yufei Tao, and Dik Lun Lee. Location-based spatial queries. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD03, pages 443–454, New York, NY, USA, 2003. ACM.

[37] Rui Zhang, Jingchao Sun, Yanchao Zhang, and Chi Zhang. Secure spatial top-k query processing via untrusted location-based service providers. *IEEE Transactions on Dependable and Secure Computing*, 12(1):111–124, Jan 2015.