MONASH University

Reverse Nearest Neighbourhood Search in Spatial Databases

Nasser Allheeib

A thesis submitted for the degree of Doctor of Philosophy at
Monash University in 2020

Faculty of Information Technology

Copyright Notice

I dedicate this thesis to my loving parents who value knowledge above all.

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief. I have duly acknowledged all the sources of information which have been used in the thesis

Nasser Ibrahim Allheeib

Signature:

Date: 29 October 2020

Publications during Enrolment

Publications arising from this thesis include:

Paper	Publication Title	
1	**N Allheeib, MS Islam, D Taniar, Z Shao, MA Cheema. Density-based reverse nearest neighbourhood search in spatial databases, In Journal of Ambient Intel-ligence and Humanized Computing, 1-12, 2018**	*Chapter 3 has been published in paper 1*
2	**N Allheeib, D Taniar, H Al-Khalidi, MS Islam, KM Adhinugraha. Safe Regionsfor Moving Reverse Neighbourhood Queries in a Peer-to-Peer Environment, In IEEE Access ,8, 50285-50298, 2020**	*Chapter 4 has been published in paper 2*
3	**N Allheeib, KM Adhinugraha, D Taniar, MA Cheema, MS Islam. Reverse Nearest Neighbourhood Queries (RNNH-RN) on Road Networks, In World Wide Web (WWW) Journal**	*Chapter 5 has been submitted for publication in paper 3.*

Acknowledgements

First and foremost, I would like to thank God for his blessing and guidance in my life, without which I could not have finished this thesis.

I am thankful to my dad and mum (Ibrahim and Latifa) who have been unwavering in their support, and have always been ready with enormous patience to catch me when I fall. Having you both behind me has made me stronger, and I totally understand being far from you has not been easy and it was also a difficult time for you, so just saying "thank you" to parents does not do justice to the way I feel.

I thank Associate Professor David Taniar, my main supervisor, who has consistently guided me, always with patience and encouragement. He directed me in this project since the first day of my PhD journey up until the submission date of my thesis. He constantly gave constructive feedback for all my work and helped me face and overcome exciting challenges. I would not have been able to succeed in this undertaking without his advice.

I also wish to express my gratitude to Dr. Muhammad Aamir Cheema, my co-supervisor, who has directed me in my PhD and has provided valuable academic and moral support. I also thank Dr. Saiful Islam, my external supervisor, who has offered me much valuable advice in regard to my work during my PhD candidature. Despite living in a different state, he willingly made time for meetings whenever he visited Melbourne and encouraged me to strive for excellence. Also, I am very grateful for my colleague in spatial databases research, Dr. Kiki Maulana, for his patience and trust in me, and for his guidance and advice which was always much appreciated.

I thank all my family members for encouraging me to study overseas and always giving me their support, in particular, my brothers and sisters in Saudi Arabia who have assisted me during the most difficult times. I thank them for their advice, prayers and support during this difficult time overseas. Thank you to my two beautiful children (Azzam and Jory) who have been very patient with their father during this PhD study. Thank you for my family and everyone who was with me in Australia and support me during my PhD journey.

Again, I would like to thank everyone who has been a part of my life throughout my PhD journey. I cannot name all of you individually, but each one of you has contributed to the person I am today.

Nasser Ibrahim Allheeib
Monash University
October 2020

Abstract

Spatial databases have become a critical part of modern applications such as the Geographic Information System (GIS), targeted marketing, and optimization of urban planning. There are a lot of applications where a particular facility needs to retrieve a group of points of interest and the points of interest within the same group need to be geographically close to each other. Unfortunately, most spatial queries are intended to retrieve the points of interest individually. In this thesis, the focus is on the retrieval of a group of points that are geographically close to each other, instead of on individual and widely dispersed points. We present efficient algorithms to retrieve a group of points, named Neighbourhood. A neighbourhood is a collection of m-chained points within the maximum distance (d) between a pair of points. We are the first to introduce the definition of neighbourhood to compute Reverse Nearest Neighbourhood ($RNNH$) Query. Results obtained from an extensive experimental study demonstrate that our algorithm is very efficient when compared to the naive algorithm or traditional spatial queries.

We extend our static RNNH algorithm for continuous monitoring of RNNH queries in cases where the queries are continuously moving. We propose a neighbourhood safe region method for monitoring continuous Reverse Nearest Neighbourhood (RNNH) queries. We use the techniques in our algorithm to efficiently compute the safe region of queries. Our extensive experimental results on datasets with different densities demonstrate the efficiency and accuracy of the proposed approaches.

Also, we are the first to present a generic algorithm for the processing of the neighbourhood on real road networks. We propose a new efficient technique for computing the neighbourhood on road networks, which we named Reverse Nearest Neighbourhood on Road networks (RNNH-RN) queries. In our study, we explain the unique definition of neighbourhood in the road network setting and we propose the RNNH-RN algorithm. Our experimental study shows that our algorithm is significantly feasible when applied to a spatial road network.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Aims and Background

1.1.1 Aims

Over the last two decades, spatial databases have attracted much interest in their now considered important applications [5]. A spatial database is defined as an optimised database that stores data which in turn defines the geographic space. Although the concept of spatial databases first emerged in 1997, recently it has become an active area of research [6] [7]. Furthermore, the rapid development of Geographic Information Systems (GIS) has enabled spatial data query technology to play an increasingly important role in real life. We can see that the spatial database has become very popular in most modern applications such as location-based social networks that employ GPS to locate users and allow them to broadcast their location and share information via their mobile devices [8]. Therefore, a strong and thorough understanding of spatial data is required. In general, a spatial query is intended to retrieve geographic objects that meet specific requirements for answering the query, and it comes as either a point of interest (POI) or a region. To retrieve the queried objects, the query can be processed in two ways: point-to-point calculation or region-based calculation [9]. In a point-to-point calculation, the query is answered by choosing appropriate objects from a dataset that can be used to answer the query.

In a region-based calculation, the query is answered by means of constructing the region that contains the correct objects that answer the query. This method returns a group of objects in the region. Region-based calculation has one major advantage over point-to-point calculation in that it does not check each point one-by-one. Consequently, this method will not suffer from degraded performance where there is a large number of objects [10]. Also, region-based calculation returns a group of

objects as a result. A Voronoi diagram is a commonly used method that applies region-based calculation. This method divides the space area into smaller spaces (cells) based on the distance to facility objects [11]. In the Voronoi cell of q, all objects inside of this cell of q consider q as the nearest facility. These objects are considered to be members of the q region. Even though a Voronoi diagram returns these objects as a group of objects (region members), there are many applications where a group of objects needs to be retrieved, and the objects in the same group need to be geographically close to each other. Reverse Nearest Neighbour queries are one type of notable solution but unfortunately, Reverse Nearest Neighbour queries and their variants are not suitable for these kinds of applications because they have some limitations: (1) the objects in the same group are sparse objects, so the distance between objects in the same group can very markedly, (2) the objects are not clustered[12]. Unfortunately, previous studies have not addressed these issues.

In this thesis, we discuss the queries in the context of a group version of Reverse Nearest Neighbour queries known as a Neighbourhood. A neighbourhood is a collection of chained objects within the maximum distance between a pair of neighbourhood members. In the neighbourhood context, instead of fetching dispersed objects as in Reverse Nearest Neighbour queries, we are interested in finding neighbourhoods of objects. The neighbourhood finds a given query as their nearest facility among all the existing facilities, and neighbourhood members within the same neighbourhood are geographically close to each other.

Therefore, we present and propose new novel queries in the context of a group version of Reverse Nearest Neighbour (RNN) queries, we named it the Reverse Nearest Neighbourhood Query that can be applied for continuous queries and in road network environment.

1.1.2 Background

Apart from the exponential increase in the acceptance of smart phones, inexpensive position locators and location-based services (LBS) are becoming increasingly popular. Consequently, the spatial database has become a critical component of modern applications. One common query that can be posed by users of applications based on location-based service (LBS) is “Find the nearest object of interest”. As an example, we consider a problem involving customers and a store. The problem concerns the delivery of online orders during COVID 19, where both stores and customers are in lock-down. A store (or supermarket) is a business that sells products to customers, and it has warehouses in various locations. The customer in this instance

is the person ordering items online and requiring home delivery. The overall cost of items is the price of each product plus the delivery cost (determined by the distance between the customers and the store). The query posed by a store (query point) is: "Find customers who are located near the store's warehouse". There are two ways to answer this question. From a facility perspective, it is important to find the important object p for a given query. We assume p is considered to be important if it is the closest point to q. This kind of query is known as k Nearest Neighbour (kNN) query in the spatial database. Given a query point $q \in F$, an integer k and a set of points P, a kNN query returns k closest points p to q, where $p \in P$.

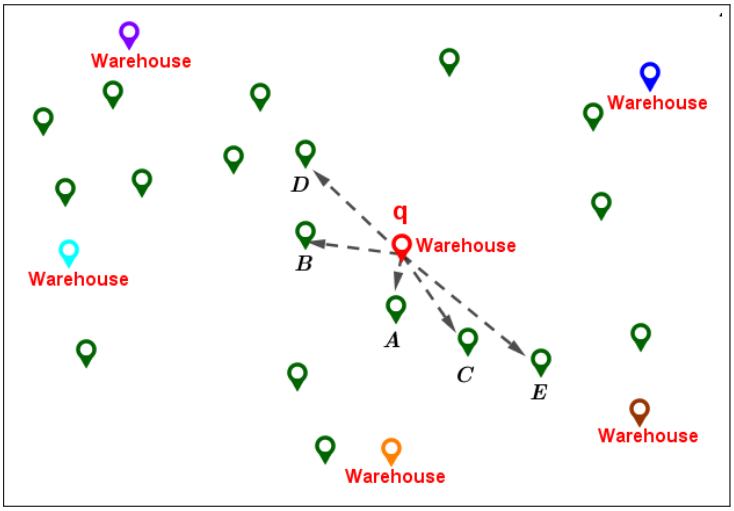


Figure 1.1: k Nearest Neighbour queries

To illustrate the idea of a k Nearest Neighbour kNN query, consider the example depicted in Figure 1.1. Assume there is a store (e.g. Coles supermarket), that has six warehouses, and the number of customers is represented by green markers. Coles (query point) is searching for customers, who have ordered online and who are located close to a warehouse. A spatial database is able to manage information about the store's warehouses and the customers, and it can provide information about the customers who are closest to the store's warehouse. In order to answer this kind of query, the entire distance between the store's warehouse (query point) and all customers is calculated. Basically, this query is solved by means of a point-to-point calculation.

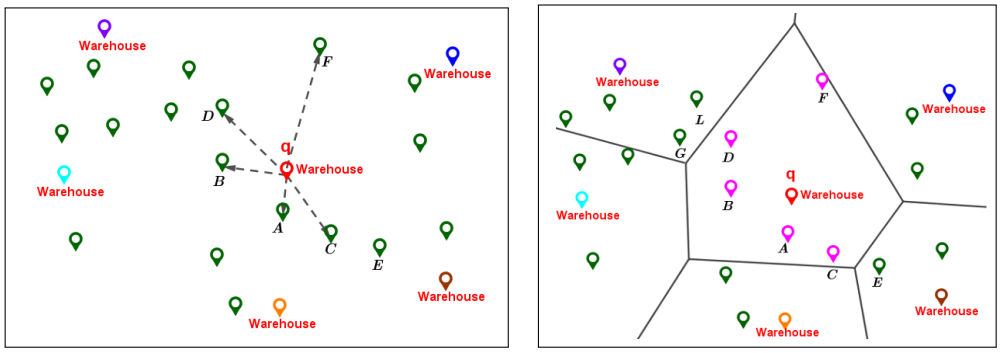
Figure 1.1 shows the nearest neighbour query for Coles' warehouse (q). In this example, the nearest neighbour is customer A, which is the closest to the warehouse (q). If the warehouse wants to obtain the groups comprised of as certain number of customers (e.g. five customers), then the kNN query based on point-to-point

calculation returns A, B, C, D and E. However, although this approach suffers from performance degradation because checking is done point by point, it also returns some points of interest that have less influence on the query, such as customer E who does not consider the query point (q) to be the nearest of all facilities. Also, the kNN query result returns customers that are dispersed geographically.

Another way that a spatial query can be addressed is from the user's/customer's perspective. This query aims to retrieve the important objects or users from the user's point of view. This approach identifies the potential objects for which the query is considered important (i.e. the closest) for them. This query is known as a Reverse Nearest Neighbour (RNN) query in spatial databases. Given a query facility q and a set of points p, an RNN query returns every user/customer $p \in P$ which considers q as the nearest of all facilities. When considering the example of a store warehouse and customers, all the potential customers closest to the store's warehouse (query) are considered. In this case, information about the potential customers of a store's warehouse can be used to make decisions or target market analysis. for example, distribution deals or promotions can target certain customers who are more likely to be influenced by deals.

The RNN query returns only those objects who consider the query point is the nearest facility of all facilities, even if it far away. Figure 1.2(a) shows an example of a RNN query-based point-to-point calculation. The RNN query of q store's warehouse comprises: A, B, C, D and F as customers who consider q as the nearest facility among all the existing facilities. It does not return customer E because there is another warehouse which is much closer to the customer E than the query point. The reverse nearest neighbour has received much research attention in recent years because it influences objects by query point. The Reverse Nearest Neighbour (RNN) query has a good advantage in that it can be answered through point-to-point and region-based calculations.

The Voronoi diagram is one of the most commonly employed methods that use to answer the Reverse Nearest Neighbour (RNN) query by region-based calculation. The Voronoi diagram of a point set F, which is donated by $VD(F)$, is a unique diagram that consists of a set of a collection of Voronoi polygons (Voronoi cells) VFs. Each Voronoi polygon is associated with a point in F (called generator point) and contains the locations of all points in the Euclidean space that are closer to the generator point of the Voronoi cell than any other generator point in F. The Voronoi cell of q is an area surrounded by the boundary in a convex polygon shape



(a) Based on Point-to-Point Calculation (b) Voronoi Diagram

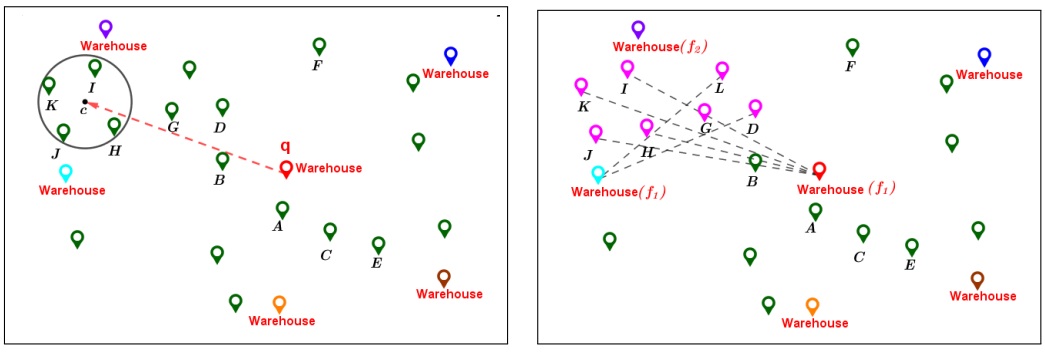
Figure 1.2: Reverse Nearest Neighbour query

where every object located in this area will always consider query point q as the nearest facility point as shown in Figure 1.2(b). Since these objects: A, B, C, D and F are located in the Voronoi cell of q, they are the answer to the $VD(q)$ query. Any objects located outside the Voronoi cell of q will not consider q as the nearest facility point. However, even though this kind of query based on region-based calculation returns points that exert great influence on the query, there are some points just outside the border of $VD(q)$, but it is better to include them as part of influence points of q such as (G and L) in Figure 1.2(b). This does not mean that these two points always move to the Voronoi cell of q and do not belong to their nearest facility. This is certainly not suggested. It means these two points belong to their nearest facility and also constitute some of the influence points for the q point. The second problem associated with the Voronoi diagram, is that points located in a Voronoi cell are dispersed objects. Thus, travelling from one point to another point is a costly computation because these points are dispersed within a particular area. In delivering orders during lookdown, the travel involved in going from one point to another point is costly because these points are dispersed.

Another approach that can be used in spatial databases to overcome the problem of dispersed objects is that instead of returning the individual single point of interest (POI), points of interest (POIs) (group objects) are found which are not dispersed; this is referred to as a Group Nearest Neighbour (GNN) query. This kind of query returns the centre of the circle that includes a group of points, and is based on the nearest smallest enclosing circle; hence, it is a geometric approach. This approach is used to find clustered points that are located close to each other geographically. This kind of query seeks a certain number of points of interest (minimum number is pre-defined by user query) that are geographically close to each other to minimise the cost of travelling between group members. A typical example of a group nearest neighbour query is "Find the group of customers closest to a query point". Consider

the example of a store and its customers. A common question that can be posed at the store's warehouse location is "Where is the nearest area that includes a group of customers (e.g. at least four)". Finding a group of points of interest (POIs) is crucial for a decision system, as it can explore and discover the surrounding clustered points of interest.

The usual search query such as the nearest neighbor query, will return the nearest customer or a sparse group of k nearest customers. However, the group nearest neighbour will return clustered customers. This is done by finding the location of the nearest group of points, and returning the centre of the circle that includes the group of points using the geometric approach of the Nearest Enclosing Circle (NEC). Figure 1.3(a) depicts an example of a group nearest neighbour, if the store warehouse (query point) wants to target a cluster of customers, it may help the store warehouse to launch promotions targeting this particular group of customers. The Group Nearest Neighbour (GNN) query returns the center of circle c that includes H, I, J and K customer. This approach aims to minimise efficiently the travel time between group members. Another spatial query has been presented is the Group Reverse kNN Query [13], it returns group of points of interest. However, Group Reverse kNN query ($GRKNN$) is must issue from a set of query points (multi-query points) to find the objects that consider the set of query points as their nearest facility points. Figure 1.3(b) shows result of $GRNN(f_1, f_2, f_3) = \{K, I, L, G, D, H, J\}$.



(a) Group Nearest Neighbour query

(b) Group Reverse kNN Query

Figure 1.3: Group spatial query

The main weaknesses of these methods are as follows: (1) both methods return only one group of objects, (2) the group that return could be located any where in space. This is because the GNN query uses the geometric approach of the Nearest Enclosing Circle (NEC); hence, it is unlikely to find a nearby group of objects within the smallest enclosing circle. Instead, it would usually be far away from the query

point as shown in Figure 1.3(a). (3) $GRKNN$ query uses more than one query point, and (4) in (GNN) query, it is unable to include very close points of interest from the boundary of circle (such as G point), the diameter of the circle is $0.85km$ in the example of Figure 1.3(a), but if we increase a little bit of the diameter to be $1.3KM$, we are able to include the G point. **Unfortunately, none of the previous spatial queries is able to handle these issues.**

Therefore, it is a great opportunity for us to propose a novel technique for solving these particular problems. Thus, instead of forming the group of objects based on the Nearest Enclosing Circle (NEC) approach or using multiple query points, we can form the group of objects in chained objects form and return numbers of group of objects result for one query point. We name the approach of group of objects as "Neighbourhood". An example of a neighbourhood is shown in Figure 1.4. Here these are two two neighbourhoods, and they are located close to the query point (q).

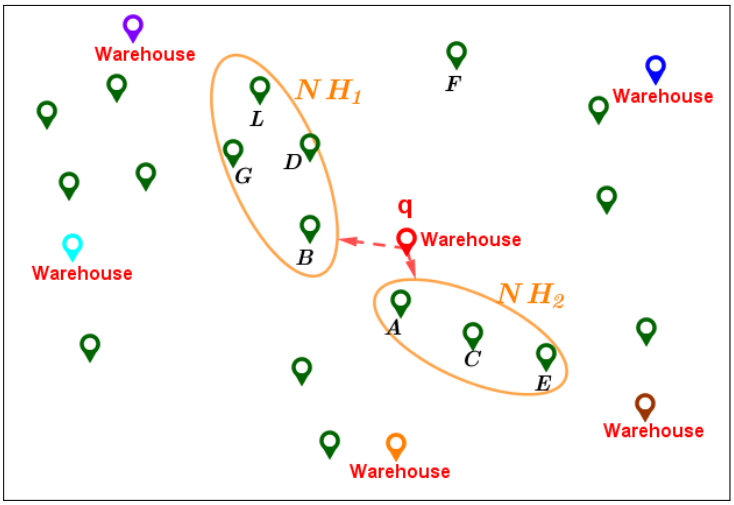


Figure 1.4: Example of chained objects

In all the previous scenarios, the query point is a static query which is one that requires the result to be computed only once based on a current snapshot of the data. Another approach that can be used for a spatial database is the continuous query. A continuous query involves points of interest that change their locations over time, and requires a higher update frequency [14]. The continuous query concept has been widely investigated over the last decade, which is why most of the current studies concentrate more on monitoring such queries [15]. Most of the studies on the continuous monitoring of spatial queries use a client and server model. Continuous queries send their locations to the server after every time unit and the server computes the result of the query accordingly and sends them back to the query point.

To illustrate the need for monitoring continuous query, let us consider a real-life example of disaster management involving bushfires which pose great danger to the affected areas, and need to be handled swiftly and reliably [16]. In Australia, during the summer months, dozens of fires burn across the country and residents brace themselves for catastrophic conditions [17]. As shown in Figure 1.5, people living in bushfire-prone areas often have to leave dangerous areas as a matter of urgency; therefore support for people in these locations must be maximised. During this kind of natural disaster, local people are disconnected from the centralized network station and communication between people can be achieved via short-range device such as Bluetooth or WiFi.

In this example there are two neighbourhoods each one with four people as shown in Figure 1.5. In one neighbourhood, neighbourhood members can communicate with each other via a short-range device. People in this disaster area can use this short-range device to contact the drone flying within its communication range to obtain information, such as the location of the nearest shelters or the latest update about the direction of fire. Drones fly around area to collect and send information to neighbourhood members. In this scenario, a flying drone is considered as a continuous object because it frequently change its location over time, which requires a huge level of updating in the drone center's database server for monitoring. Also, when the drone flies long distances and is far away from its current location, this could change the query point (flying drone) and requires the allocation of a new neighbourhood by the server.

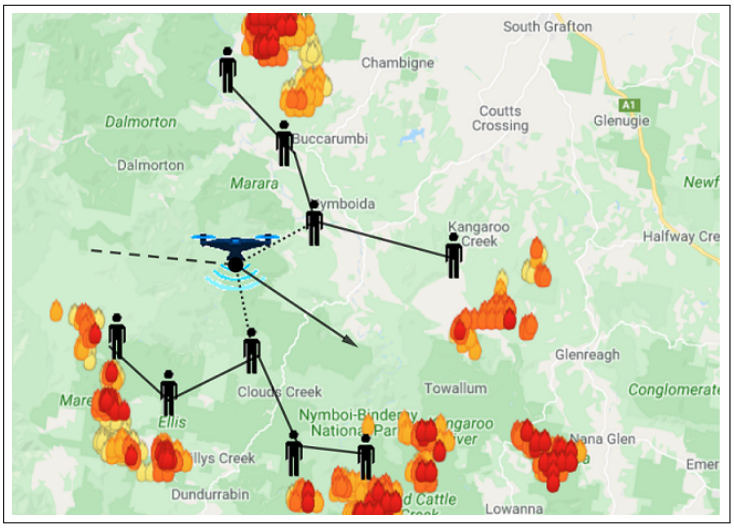


Figure 1.5: Continuous monitoring query

Monitoring the continuous query plays a significant role in query processing, as

results must be kept up to date. There are two main challenges in monitoring the continuous query efficiently: the first is to reduce the computation cost. The query results may need to be recomputed whenever a query point changes its location. This may result in a huge computation cost for every single new query point location. The second challenge is to reduce the communication cost, since the query point is required to report its location to the server every time it changes its location. So, the aim is to present a comprehensive approach to the monitoring of continuous spatial queries. Over the last few years, several safe-region-based approaches [18, 19, 15] have been introduced to monitor various continuous spatial queries. The aim of the safe region algorithm is to return an area or zone such that the results of the query remain unchanged as long as the query remains inside the safe region. Hence, the results of the query do not need to be updated unless the query leaves its safe region.

To the best of our knowledge, all these previous approaches concentrate on traditional queries such as kNN, RNN and range query; none of the works is concerned with the continuous query in the context of a neighbourhood version of a neighbourhood version of RNN queries.

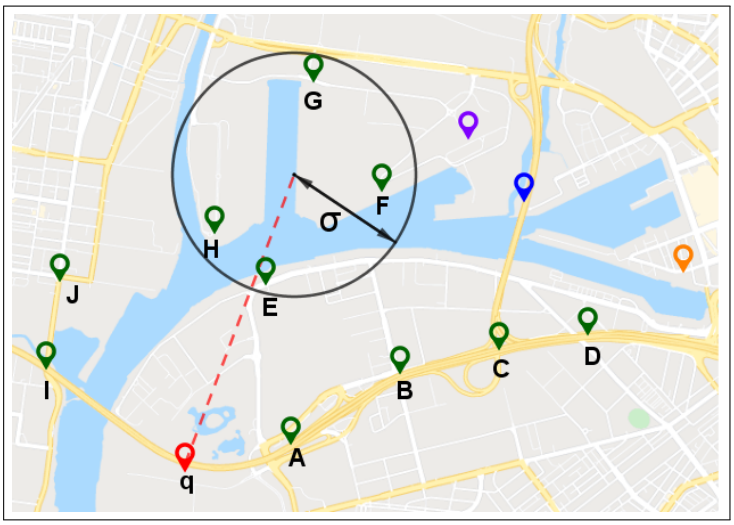


Figure 1.6: Example of euclidean group nearest neighbour query

In spatial databases, there are two different types of environments: Euclidean space and road networks. All previous studies are concerned with Euclidean space. However, to obtain a reliable result, the road network approach is more realistic, because Euclidean space is not always accurate. The problem becomes more complicated when we use networks distance. To illustrate this point, consider the example in Figure 1.6, where the query of Group Nearest Neighbour $4GNN$ returns the group nearest neighbour that consists of four points (E, H, F, G) in Euclidean space.

However, in reality, these points of interest are not the nearest neighbours to the query, because the query would have to travel around the river and the buildings to reach these points (E, H, F, G). It is very clear that the distance measurement in Euclidean space depends only on the relative positions of two points of interest, whereas the real distance measurement on a road network does not depend solely on the relative positions, but on the road links (sections) between the two points of interest while taking into consideration any obstacles such as rivers and/or buildings.

Recently, spatial queries on road networks have been widely studied, the most commons works being on Nearest Neighbor (NN), k Nearest Neighbor (kNN), Continuous Nearest Neighbor CNN , and Reverse Nearest Neighbor (RNN) query [20]. However, **all these previous studies focus on finding, processing and retrieving the points of interest individually on a road network; no work has attempted to find the points of interest on a road network by considering a neighbourhood.**

1.2 Problem Definitions

In this study, our aim is to address the shortcomings of current methods by proposing a group version of RNN queries as a neighbourhood. This research studies and proposes a novel approach called Reverse Nearest Neighbourhood query where a neighbourhood considers the query as the nearest facility. Given the limitations of the approaches described above, this thesis will focus on the closeness of the neighbourhood members (clustered points), and minimise the distance between them. Another aim is to minimise the distance between the query and neighbourhood. To accomplish this goal, it is important to understand several research challenges:

- **Challenge 1: Limitation in the Definition of Neighbourhood**

  There are many applications where the query needs to retrieve a neighbourhood whose members need to be geographically close to each other and close to the query point. One good example would be targeted marketing, whereby the query can design special promotion plans or deals for customers within a certain neighbourhood. Members in the neighbourhood are not far from each other geographically, and the neighbourhood is not far from the query. The query could minimise delivery costs when offering promotions to customers to help increase revenue. Hence, the first challenge and one that is the most crucial, is how to define the neighbourhood that meets the given description.

- **Challenge 2: Expensive Cost of Finding Neighbourhood**

  The existing methods applied to Reverse Nearest Neighbour queries cannot retrieve a neighbourhood. To the best of our knowledge, to date, no algorithm has been proposed for solving Reverse Nearest Neighbourhood queries. Unfortunately, the baseline algorithm that can be used for neighbourhood queries is computationally expensive. Therefore, the next challenge is to reduce the cost of finding a neighbourhood.

- **Challenge 3: Reducing the Computation and Communication Cost**

  In continuous queries, the query results may need to be recomputed whenever a query changes its location. This may incur a huge computation cost. Also, to maintain the correctness of the query results, a query is required to report its location to the server every time it changes its location, in which case the communication cost may increase significantly as well. Therefore, the challenge here is to reduce the computation and communication cost incurred by the monitoring of continuous queries.

- **Challenge 4: Creating the Neighbourhood in a Spatial Road Network**

  Many studies on spatial databases focus on Euclidean space and road networks. In real life, the Euclidean distance-based approach is not always accurate [21]because it ignores obstacles such as buildings or a lake. However, in road networks, obstacles are considered. Therefore, the last challenge for this study is to design a neighbourhood for a road networks environment.

1.3 Research Objectives

To address the aforementioned research challenges, this section states the primary aim which is to propose efficient techniques to solve Reverse Nearest Neighbourhood queries. In order to achieve this key goal, we establish specific objectives, formulated as research questions.

- *RQ1 - How can the Reverse Nearest Neighbourhood $RNNH$ queries be better captured and efficiently computed?*

This work is motivated by our observation that RNN may fail to retrieve a neighbourhood of objects. We show that RNN query may be unable to retrieve neighbourhood of objects that are geographically close to each other. To better capture the notion of neighbourhood, we aim to return a neighbourhood that considers the query is the nearest facility and neighbourhood members need to be geographically close to each other. This research question addresses the first and second challenges and published in [22], more details of which are presented in Chapter 3.

- *RQ2 - How to efficiently monitor continuous Reverse Nearest Neighbourhood ($RNNH$) queries?*

  In this work, we aim to monitor $RNNH$ query in a scenario where the query is continuously moving and users' locations are stationary. We propose a safe-region technique to monitor queries with a minimal computation cost. Therefore, the third challenge is addressed by this research question and published in [23]. More details are presented in Chapter 4.

- *RQ3 - How can the Reverse Nearest Neighbourhood queries on road networks be computed?*

  This work is also motivated by our observation that $RNNH$ queries cannot be directly implemented into a spatial road network and $RNNH$ fails to find accurate Reverse Nearest Neighbourhoods on road networks. Euclidean distance does not take obstacles into consideration, and therefore it is totally inappropriate for road networks. Consequently, it is important to have an efficient algorithm that is applicable to road networks. The fourth challenge is addressed by this research question, more details of which are presented in Chapter 5.

1.4 Contributions

In this section, we summarise the contribution that this thesis makes to this body of knowledge. Each of the above-mentioned research objectives are detailed below.

1.4.1 Reverse Nearest Neighbourhood Queries

Currently, we are experiencing huge volumes of spatial objects data in many applications such as mobile social networks, and the location-based mobile community can be typically formed by a group or neighbourhood comprising mobile subscribers. An understanding of these spatial data is crucial for the locations-based service providers. Thus, the trend emphasises the need to understand the neighbourhood instead of an individual object; however, firstly it is essential to define the neighbourhood.

Hence, we formally define a Reverse Nearest Neighbourhood query in a spatial database, and propose a novel query called Reverse Nearest Neighbourhood ($RNNH$) query for two-dimensional location data. That is, instead of retrieving dispersed objects as occurs with RNN queries, we find a neighbourhood of objects which find a given facility which is the nearest one of all existing facilities. A neighbourhood is a collection of chained objects within the maximum distance between a pair of neighbourhood members. The neighbourhood members are influential objects for the given point that are clustered and not sparse objects. Also, the objects returned by a Voronoi diagram in a RNN query are not the only influential points for the query point [24]. In fact, other objects can exert a great influence on the given query point.

As an example, we consider a problem involving customers and a store in a scenario where online order is being delivered. The problem arises when several customers are ordering items from the nearest store and each customer is paying individual delivery fees. The store might be able to reduce the delivery cost by grouping under a single dispatch the orders of all customers who are close to each other. A store can take into consideration the price of the goods and the distance between the customers in the delivery cost. As discussed, the RNN query using a Voronoi diagram can return B and D customers as chained objects in one neighbourhood that consider q as the nearest facility. However, we can see that the best option for L and G customers is to join with customers (B, D) in a nearby group in one neighbourhood to reduce the delivery costs as shown in Figure 1.4. Therefore, in this case, the store's warehouse (query point) will be able to offer the lowest delivery cost for the neighbourhood of B, D, L and G. Our extensive and comprehensive experiments on synthetic and real datasets demonstrate the effectiveness and efficiency of our proposed algorithm.

1.4.2 Continuous Reverse Nearest Neighbourhood Queries

In this study, we extended our work on snapshot $RNNH$ queries for the continuous $RNNH$ queries. The moving query is assumed to be constantly moving. Minimising the frequent updates of the query location and keeping costs low while monitoring the moving query are the two main challenges for researchers. These challenges are addressed by using the safe region concept. The safe region is an area where the result of query does not change as long as the query remains inside it.

The aim of the safe region is to reduce the number of query location updates to the server. The query does not communicate with the server once it enters the safe region, and even the server is not aware of the direction of the query. It reports to the server only when the query leaves its safe region or when the server requests it. In this thesis, we present our technique using safe region for monitoring $RNNH$ query, which is known as the $RNNH$ safe region. The aim here is to prevent any communication between the query and server while the query is moving inside its safe region. We conduct extensive experiments to demonstrate the effectiveness and efficiency of our approach.

1.4.3 Reverse Nearest Neighbourhood on Road Networks Queries

During the review of literature on Reverse Nearest Neighbourhood queries, we noticed that the efficacy of this algorithm has not been investigated in the context of spatial road networks. The algorithm of Reverse Nearest Neighbourhood in Euclidean distance cannot be implemented directly in spatial road networks, as the characteristics of a spatial road network differ from those of Euclidean distance. When computing distances, algorithms applied to Euclidean distance ignore the impact of obstacles such as buildings and lakes. Take as an example a city center with a river and buildings (off-roads space) as shown in Figure 1.7. The algorithm in Euclidean space returns one neighbourhood that has a collection of chained objects with a maximum distance between a pair of neighbourhood members of two km. As shown in the Figure, the neighbourhood has A, B, C and D objects. Unfortunately, Euclidean space does not take into account obstacles such as the river in this example. The actual distance between object B and C,is not within two km; it is more than ten km. Realistically, objects c or d would have to travel around the river to get to q. Hence, we found that the distance in Euclidean space depends solely on the relative positions of two points, while the distance in a spatial road network depends not only on relative positions, but on sections of road between two points.

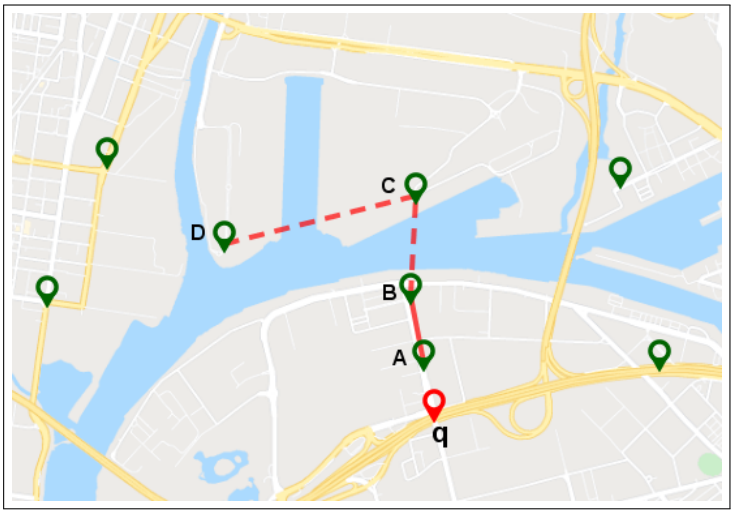


Figure 1.7: Example of obstacles in spatial databases

Therefore, in this study, we introduce an algorithm which takes obstacles into account. We propose the Reverse Nearest Neighbourhood on Road Networks ($RNNHRN$) queries for finding neighbourhoods in a road network. To the best of our knowledge, this is the first study to propose an algorithm for retrieving a neighbourhood on road networks. Road networks are usually represented as a graph (a web of roads) that consists of edges and vertices (nodes). An edge is a line that links two vertices or nodes. Computing the distance between two nodes in a road network involves calculating the distance of edges that link two nodes. We conduct extensive experiments to demonstrate that the proposed solution is applicable for network datasets with low, medium and high levels of population density.

1.5 Thesis Structure

The structure of this thesis is given below.

- Chapter 2 reviews the related studies that motivated us to conduct research on reverse nearest neighbourhood.
- Chapter 3 describes our work on spatial Euclidean distance, where we propose and define the concept of a neighbourhood in Reverse Nearest Neighbour Query, which we have called Reverse Nearest Neighbourhood ($RNNH$) query.
- Chapter 4 presents a new method for monitoring continuous reverse nearest neighbourhood queries. The moving query requests the constant reporting of its results which extend from the registration of the query to its cancellation.

Over this time, the query results must be continuously updated according to its location. So, we construct a new technique to reduce the need for continuous monitoring of the query, and eliminate the need for the query to follow a defined path.

- Chapter 5 describes our examination of reverse nearest neighbourhood queries to road networks. Since a road network is generally represented as a graph, finding the neighbourhood in Euclidean distance is not the same as in road networks. Consequently, it is important to devise an algorithm for Reverse Nearest Neighbourhood on Road Networks, known as the $RNNH-RN$ query.

- Chapter 6 summarises the outcomes of our studies and concludes our research. This chapter also includes several suggestions for possible directions for future work

Chapter 2

Literature Review

2.1 Overview

Presented in this chapter is a review of the relevant studies, so that we obtain a good understanding of some of the key elements in the spatial queries field. We start by discussing and providing background information on query processing that is based on Euclidean space. We discuss the fundamental spatial queries, range query, nearest neighbour query, reverse nearest neighbour query and group nearest neighbour query. Furthermore, the related work concerning the monitoring continuous spatial queries is discussed, and an overview of the relevant work on the safe region is presented. We review several relevant studies dealing with spatial queries based on road networks. Finally, we summarise the limitations of the studies that have been reviewed.

2.2 Spatial Query Processing

2.2.1 Range Queries

A range query in spatial databases is used to find and retrieve objects of interest within a given radius. In this research, we focus on the static radius range query (for the sake of simplicity we name it 'range query') which is applied when the query is not moving. A range query depends on the current location of the query, and it can be defined as follows: Given a set of facilities F, a query facility $q \in F$, a set of points P and a positive value d, a range query retrieves all points that lie within distance d from q. Formally, a range query returns every point p for which $dist(q, p) \leq d$, where $dist(q, p)$ represents the distance between q and p.

Range queries have recently been investigated in various environments: Euclidean space and road networks [25, 26, 27, 28, 29, 30], and their results are computed where the static location of the points and facilities is taken into account [25, 27, 29]. In Euclidean space, the range query depends on the relative positions of two points of interest to calculate the distance. Figure 2.1 below gives an example of a range query in Euclidean space, while the points of interest (i.e., restaurants) are represented by the green marker. The customer (query point) wants to know all restaurants within $2km$ from where he/she is standing. The highlighted points of interest shown in red represent the restaurant location that will be obtained by the user (listed by numbers 1 to 7), and the highlighted points of interest shown by the green marker represent restaurants outside the range that will, therefore, not be included in the final result.

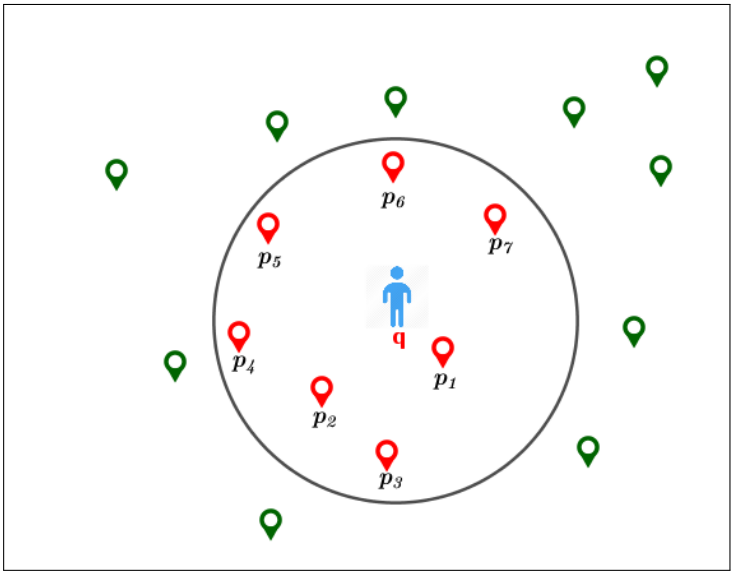


Figure 2.1: Example of a range query

Most range query processing depends on Euclidean distance to provide the relative position of the spatial object [31]. However, in some circumstances, the location of spatial objects may need to be specified by the underlying network and not by Euclidean space. Thus, researchers [32, 33, 34, 35] investigated the concept of range query using network distance. The aim of this query is to find points of interest which are based on the distance of the road network, not Euclidean distance. To give an example of a range query in a road network, consider Figure 2.2 where the points of interest (i.e., restaurants) are listed from 1 to 17. The customer's request is for all restaurants within $1.5km$ from where he/she is standing. The red objects represent the answer to the range query in the road network because their network distance is less than $1.5km$ from the location of the query point.

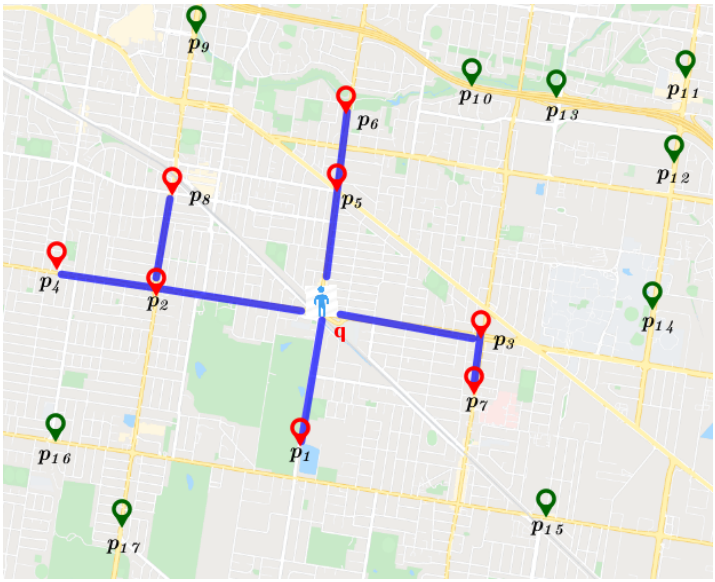


Figure 2.2: Range query in a spatial network database

Most literature on range query is based on R-tree proposed [1] in 1984 and and consistently applied due to its efficiency and good performance. R-tree indexing can index multidimensional objects. It is based on their closeness and binds them in a minimum rectangle. The processing of a range query using an R-Tree index is an efficient method. [1] demonstrates the processing of a range query using R-tree, as shown in the example in Figure 2.3. The range query typically represents a circular area with a specific radius [25], a query point is the centre of the circle and all objects falling within that area would be the result of the range query. The R-tree traversal starts with the root to answer the query range. First, entries that overlap the range are retrieved (in this example they are $E1$ and $E2$). Then, it expands from the root to all the entries that overlap the query range (E_6). Entries that do not overlap the range (e.g., E_3, E_4, E_5) are skipped. This process is repeated until it retrieves all of the leaf nodes that have equal or less distance d from the query point[25].

2.2.2 Nearest Neighbour Queries

In the last two decades, the problem regarding the nearest neighbour query has been the subject of significant research attention and especially in terms of data analysis and information retrieval [36]. Nearest neighbour query is motivated by the importance of using these queries in many fields such as geographical information systems (GIS) and learning theory. The k-nearest neighbour (kNN) query is a method of finding the k closest objects to the query point, and the best first search

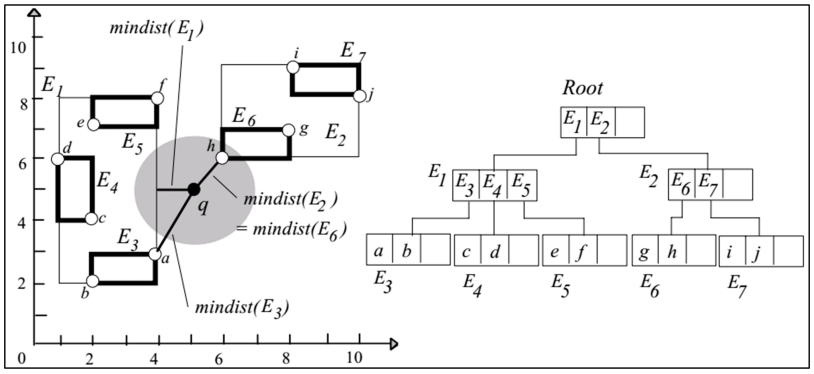


Figure 2.3: Range query with R-tree [1]

algorithm was proposed by [37] and subsequently refined by [38]. kNN queries have been studied in a variety of environmental settings such as Euclidean space, where the distance is measured as the straight line distance between two objects [39, 40, 41]: firstly, in road networks, where distance is the length of the shortest path between two objects' locations [42, 43, 44, 45, 46, 47, 48]; secondly, in indoor space, where distance measurement considers indoor entities, such as rooms and doors, which enable and constrain indoor movement [49, 50, 51, 52]; and thirdly, in obstructed space, where distance is measured as the shortest path connecting two objects without crossing any obstacle [53, 54, 55, 56]. Specifically, we highlight the k Nearest Neighbour (KNN) query in Euclidean space and road networks environments. The the k Nearest Neighbour (KNN) query in Euclidean and road networks space has been widely applied in many fields including spatial databases and been investigated in various settings, such as static queries, where the results are computed based on the stationary locations of the objects [14, 57, 58, 39, 59, 60, 61, 41, 62, 63]. It has also been implemented for continuous queries, where the query results are continuously monitored [64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76].

In a spatial database, there are a number of common index structures used for K Nearest Neighbour queries: R-tree [37], SR-tree [77] which is the enhancement of R-tree, PK-tree [78] which is based on quadtree, and PK+tree [79] which is an enhancement of PK-Tree. The most common kNN algorithm search is done using R-tree. R-tree groups the spatial objects based on their closeness and binds them within a minimum rectangle. The grouping continues until the top level consists of a single root. It prunes unnecessary candidates in order to reduce the processing time. When searching, the traversal on R-tree starts from the root in a depth-first manner to retrieve points of interest [38].

To illustrate the processing of a kNN query, consider the example depicted

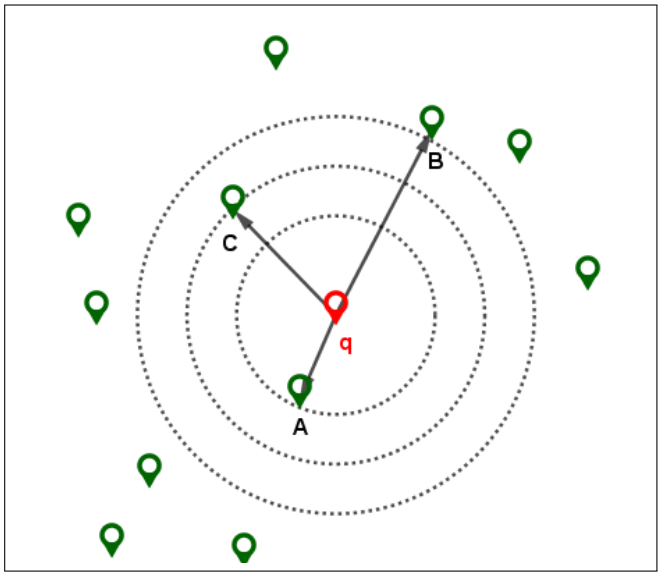


Figure 2.4: Example of a 3NN query

in Figure 2.4. In this example, the query point q is looking for the nearest three neighbours, so query issues the k Nearest Neighbour (KNN) query where $k = 3$. In kNN query, the main aim is to find the three objects that are nearest to the query point where the Euclidean distance is used as the metric. The Euclidean distance between an object and the query point is indicated by a circle with q as the centre. From this example, A is the first nearest Neighbour ($1NN$) of q, C is the second nearest neighbour of q and B is the third nearest neighbour of q. Hence, the nearest objects three to q are A, B and C.

A road network is another common space model in the spatial database field [80]. In Euclidean space, the distance between two objects is a straight line. However, this is not the case in road networks, where the distance between two objects is restricted to the shortest path as a real road network. [32] was the first author to introduce the nearest neighbour query on road networks. Incremental Euclidean Restriction (IER) [32] is a kNN method that uses a simple Euclidean distance heuristic. IER retrieves Euclidean kNNs as candidates and computes network distances to each one using a shortest path algorithm. Recent survey experimental work for KNN on road networks has been undertaken [80], the main finding of which was the surprising performance of IER and the implications this has for heuristics used in kNN query.

In particular, spatial databases represent the spatial data in the road network in graph G, and the graph consists of vertices and edges, where each vertex is an object and each edge is a road or link from one intersection to another. An object

in a spatial road network can be a road intersection, curved road, connected and disconnected roads or object location [81]. We highlight the underacted weighted graph. In our road network study, the graph is denoted by $G = (V, E)$, where V is a set of vertices and E is a set of edges. The set of points of interest comprises $P = \{p_1, p_2, ..p_n\}$, a set of facilities $F = \{f_1, f_2, ...f_k\}$, and a query point $q \in F$ that reside in G. Each edge $(v_i, v_j) \in E$ has a weight (distance travel) [82], which is a positive value. If given a source vertex v_i and destination vertex v_j, the sum of weights of edges along the path is called the distance of the path in the road network, and it is denoted by $dist(v_i, v_j)$. The distance between two objects in a road network is calculated by computing the shortest paths. The Dijkstra algorithm [83] is a commonly applied algorithm for computing the shortest path in a graph, and it is undeniably one of the best methods for solving the shortest path problem, the Dijkstra algorithm has been used for decades, and many applications today are adopting it, including Google Maps [84]. Figure 2.5 depicts a road network. Every vertex or node represents a point of interest. The red nodes represent facility points and the green nodes are users/customers. The weight of the edge (P_7, P_9) is 2. Thus, the shorter path from q point to f_3 is through the path of $\{P_1, P_4, P_5\}$, which is denoted by $dist(q, f_3) = 5$, i.e. $dist(q, f_3) = (1 + 2 + 1 + 1)$. In other words, the shorter path from f_4 to f_3 is calculated thorough the path of $\{P_4, P_5\}$, and not from node p_{13}, because the distance is shorter through this path $\{P_4, P_5\}$.

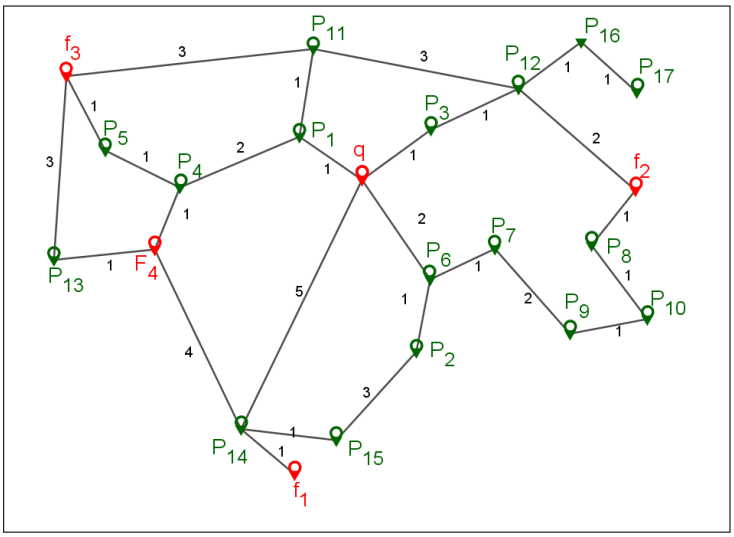


Figure 2.5: A Road Network

2.2.3 Reverse Nearest Neighbour Queries

One of the most common and fundamental queries in spatial databases is Reverse Nearest Neighbour (RNN) search [85, 34, 86, 87], which has been extensively studied [88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99] ever since it was introduced in [100]. It has proven to be useful for various purposes, such as decision support systems, profile-based marketing, referral systems, and maintenance of document repositories [80]. The Reverse Nearest Neighbour (RNN) query is a method used to retrieve the objects closest to the query point from the object's perspective. Given a set of facilities F, a set of points p as users, and a query point q where $q \in F$, Reverse Nearest Neighbour (RNN) returns every point p in P which considers q as one of the closest facilities.

The RNN query can be processed using two approaches: point-to-point and region-based. The RNN query based on the point-to-point approach was introduced by [101], where he presented the concept of the Influence Set of RNN Queries. Figure 2.6 shows an example of RNN query, it is shown searching for a set of customers that consider the supermarket for example, ALDI -query point- as the closest facility based on where customers are located. In this situation we need to compare the distance between the location of each customer to their nearest facilities. If each customer is closer to the ALDI supermarket compared to other facilities, this is a result of RNN query. In this example, the RNN result set for query point (ALDI) is (A and D) customers. Computing the point-to-point approach is in fact quite expensive.

The region-based approach is another method that has been proposed for answering RNN queries [10]. Several studies have been conducted on spatial databases for RNN using the region approach, such as the Voronoi diagram and Influence Zone. The Voronoi diagram has been widely used for processing spatial query [35, 35], specifically to answer an RNN query [12]. The Voronoi diagram works by dividing the data space into some regions; each region has a generator point where any objects inside the corresponding region consider the generator point as their closest generator point compared to the points in the other regions. Several previous studies have used the properties of a Voronoi diagram to solve the RNN query (see [102] and [103]). If a generator point in this diagram is considered as the query point, then the Voronoi cell for this generator point is considered as the region of RNN query and all objects within this region will be considered as the answer to an RNN query. The concept of Influence Zone which was introduced by [24] is also used to solve RNN queries. The Influence Zone approach finds the influence area of each query object, and all objects within this area will always see the query object as their nearest neighbour.

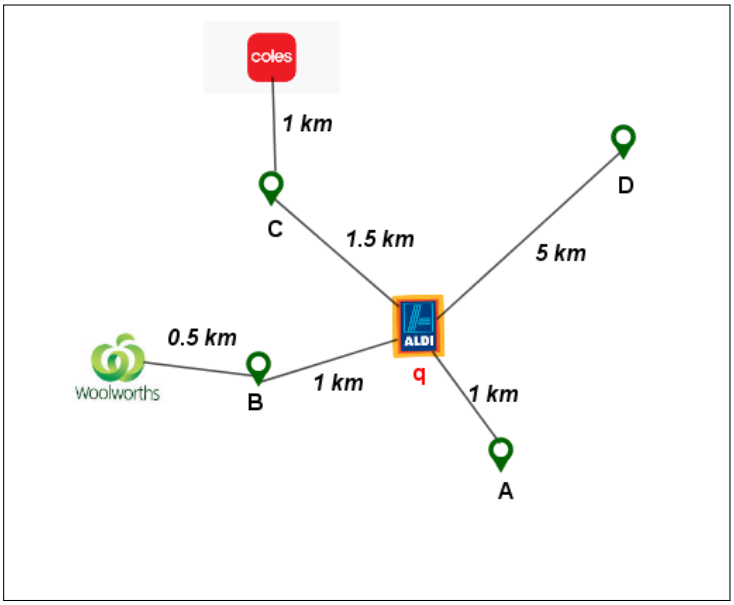


Figure 2.6: Example of RNN query

The Influence Zone computes the Influence Set efficiently, where a whole region is defined, rather than each object being verified one by one. The Influence Zone is created simply by having a perpendicular bisector at every point in the facility set (F) with the query point. Below, Figure 2.7 shows an example of an Influence Zone and Influence Set.

In general, the Reverse Nearest Neighbour query has been investigated in various settings: static queries, moving queries, Euclidean space and on road networks. Static RNN queries have been widely used in many applications and are introduced in two ways without pre-computation and pre-computing [100]. Some of the most notable algorithms and techniques have been proposed without using pre-computation to solve the static RNN query: Six-regions [91], TPL , TPL ++ [93], FINCH [92], SLICE [99] and InfoZone [24], are all algorithms based on an R-tree index. Every algorithm has a different technique for processing and addressing RNN queries, and each algorithm also processes RNN in two phases: the filtering phase and the verification phase [104]. In the filtering phase, each algorithm should use the facilities set to filter the search space that cannot be part of the result of RNN query. Hence, the algorithm does not filter more search space because this entails an extensive search, and a costly one at that.

In contrast, the algorithm that filters a larger space search can obtain results more quickly. Two pruning techniques can be used for filtering: Region-based [91] and Half-space pruning [93]. The Region-based technique requires less computing

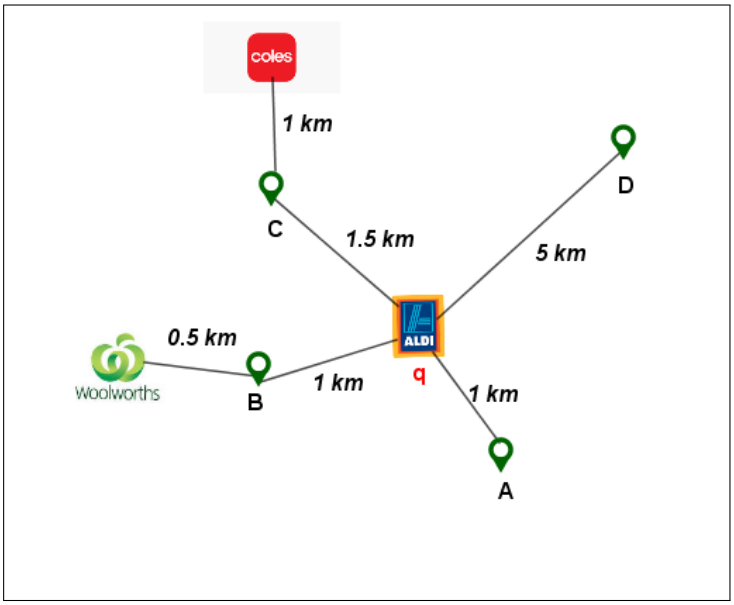


Figure 2.7: Example of the concept of influence zone

time, but the Half-space technique can prune a larger area. In the verification phase, after obtaining the candidate points of the RNN query, this phase is responsible for confirming whether or not this point can be an RNN result.

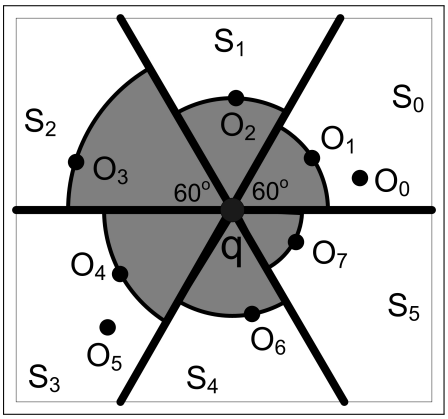


Figure 2.8: Six-regions algorithm [2]

Six-regions is the first technique proposed for solving RNN query and it does not require any pre-computation; nor does it rely on the Region-based technique. It was introduced by [105]. The six-regions algorithm [105] defines the pruning area by dividing the whole data space centred around query point q into equal regions of 60 ° (S_0 to S_5) as shown in Figure 2.8.

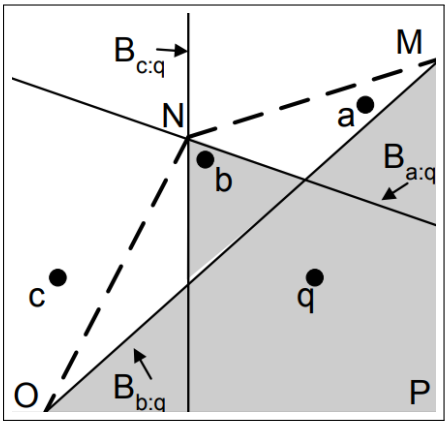


Figure 2.9: TPL [2]

The TPL technique is the first algorithm to be based on the concept of Half-space pruning for computing RNN query [106]. This algorithm is created using a perpendicular bisector between the q and F facilities point. Consider the example given in Figure 2.9, where a bisector between q and a is indicated $B_{a:q}$ which divides the space into two half-spaces. The Half-space that contains a is denoted as $H_{a:q}$ and the Half-space containing q is denoted as $H_{q:a}$. Any point that lies within the Half-space $H_{a:q}$ is always closer to a than to q and cannot be the RNN for this reason. In the containment phase, TPL retrieves the points of interest within the unpruned area by traversing an R-tree that indexes the locations of the points of interest.

The TPL++ [107] is the most current method proposed for solving RNN queries. It utilises and enhances the TPL strategy by adding optimisation features. It improves filtering in TPL $O(km)$ to $O(m)$. As in TPL, FINCH [108] uses Half-space pruning to define the pruning area. However, instead of using facility objects to filter the entries, it employs a convex polygon that approximates the unpruned area. Any object that lies outside the polygon can be pruned. The FINCH algorithm approximates a convex polygon to avoid expensive subset filtering. Clearly, the containment checking is cheaper than TPL.

The SLICE technique [107] applies the region-based technique as a filtering strategy for solving RNN query. Unlike the six-region approach which always has six partitions, SLICE divides data space into arbitrary partitions as shown in Figure 2.10. The pruning strategy of SLICE is more powerful than that of the six-region because the SLICE algorithm can prune a much larger area.

The INFZONE technique represents an improvement of the technique employed in FINCH, and was introduced by [24]. The INFZONE technique uses the Half-space

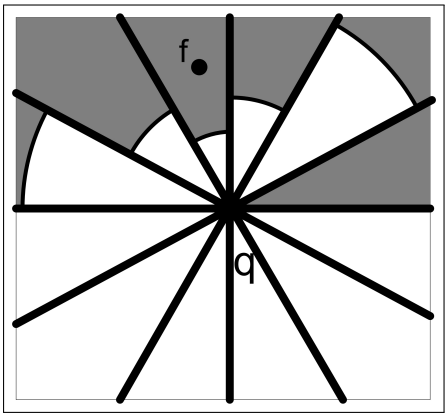


Figure 2.10: SLICE prunes larger space with more partitions [3]

approach, and generates an Influence zone where any objects within this zone are guaranteed to be the result of RNN query. It is generated based on bisector lines, where the bisector line divides the area into two equal segments called Half-spaces [10]. The INFZONE was devised to improve the verification in RNN query. FINCH differs from the InfoZone in that it has no verification phase. In other words, once the influence zone is computed all points which lie within the InfoZone can be the answer to the RNN query.

Continuous RNN queries in spatial queries have been studied extensively in recent research such as [57], [14] and [64]. Since the continuous monitoring of moving queries has a significant role in studies related to spatial databases, many scientists such as [109], [65] and [110], have investigated the monitoring of continuous objects . The first work to present the continuous RNN was [111], which assumed the objects' speeds were known. Although many studies have extended this approach, they did not consider the motion patterns of the object [24]. [112] and [113] were the first to address continuous RNN query based on six-regions and a TPL algorithm has been proposed without considering objects' speeds. The solution given in [112] is based on the approach of the six 60° regions to monitor RNN query results, while [113] proposed a monitoring algorithm called TPL for continuous RNN query based on a bisector approach. [114] and [97] proposed a strategy monitoring for continuous RNN. The concept of Lazy Updates was introduced by [97], and it is the best known algorithm for continuously monitoring RNN queries. Lazy Updates not only reduces the computation cost but also significantly decreases the communication cost.

In addition, the Reverse Nearest Neighbour query can be implemented in a road network environment [115]. As mentioned earlier, a spatial road network has settings that differ from Euclidean space, so spatial road networks can be formally represented

by a graph in RNN queries. Compared to KNN queries, in the literature, there are still a few ways to find RNN on a road networks [116]. One of the most common methods employed to find RNN query in road network is the Network Voronoi Diagram (NVD)[12], which was undertaken by [103, 35], and [117]. The Network Voronoi Diagram (NVD) is utilised to answer continuous queries on road networks [118] and, generally, it has been created using the actual network distances, not the Euclidean distance between objects. A formal definition of Network Voronoi Diagram (NVD) is found in [119] and [39]: "A Network Voronoi Diagram, termed NVD, is defined as graphs and is a specialization of Voronoi diagrams, where the location of objects is restricted to the links that connect the nodes of the graph and the distance between objects is defined as their shortest path in the network rather than their Euclidean distance", Figure 2.11 shows an example of Network Voronoi Diagram based on Network Distance, where the "Xs" are the generator points.

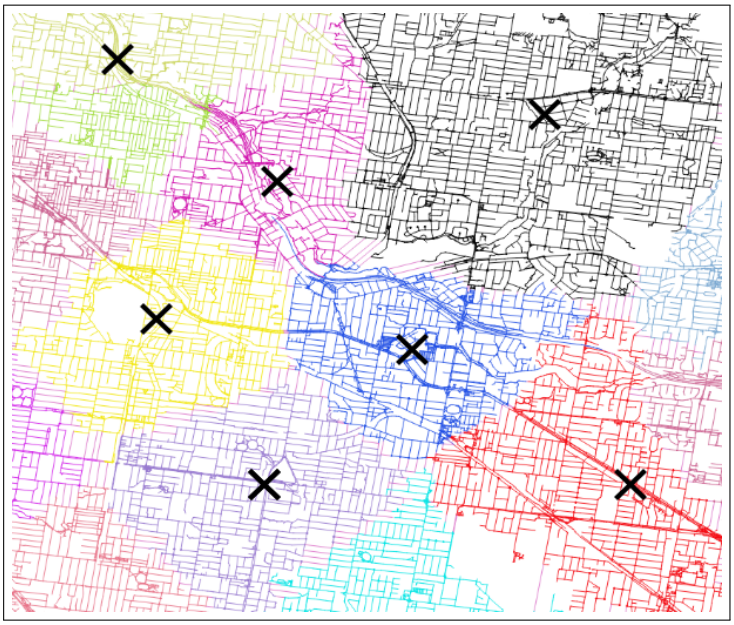


Figure 2.11: Voronoi diagram in spatial road network

2.3 Reducing Communication Cost for Continuous Queries

In spatial databases, continuous monitoring queries have been investigated in regard to various types of query, such as Nearest Neighbour (NN) query [120, 121, 75, 122, 123, 124, 125, 126], range query [127, 128, 129, 130, 131, 30] and Reverse Nearest Neighbour (RNN) query [71, 73, 132, 133, 108, 3, 134]. Referring to continuous

query, the problem of dealing with data that are continuously changing is based on the position of continuously moving objects. Most existing works on the continuous monitoring of spatial queries focus on reducing computation cost, yet the reduction of communication cost has not received much attention. Some works that concentrate on reducing communication cost include [129], [135], [136], [137] and [138]. [129] introduced MobiEyes to reduce the computation and communication costs of range query monitoring.

Another generic framework for the continuous monitoring of spatial queries is known as " safe region". The concept of safe region can slash communication and computation costs. In a safe region, a query point does not need to be connected to the server. It needs to update its location only upon leaving the safe region. Several studies such as [139] and [15] have identified various types of safe regions . [15] presents the safe region-based approach in an effort to monitor the moving skyline queries in Euclidean space, while [140] addresses the safe region concept on road networks for finding the safest paths. Additionally, [141] addressed the safe region concept by reporting the query locations to the server after t time and d distance. The study assumed that d and t units will establish the parameters of the safe region, thereby reducing the communication overhead between moving clients and the server. [142] examined the safe region concept, but limited it to a rectangular safe region that is not appropriate for a moving, circular safe region. Although the first study to address the circular safe region issue is [143], it omitted the calculation of area, as the server was in stand-by mode at a certain distance. Finally, [144] investigated circular safe regions for range queries using the Monte-Carlo method, mainly due to the irregular shape of the safe region. However, it should be noted that this method can be applied only to the range of moving queries.

Assume in Figure 2.12 that a person or customer wants to know the closest two fuel stations as he/she moves along. In this scenario, the customer (at position q) poses a query to the server returning points p_1 and p_2 as the nearest two fuel stations. After processing the query, the server returns the answer to the customer. The reason for introducing the safe region is to keep this result the same as long as the customer remains within a certain area around the initial position, which we refer to using a shaded area. In addition to the query result, the server has to return the dimensions of the query's safe region, so that consequently, there is no need to contact the server side as long the query moves inside the safe region. Thus, the implementation of a safe region has advantages in terms of cost, and it reduces the communication and computation costs between the server and clients.

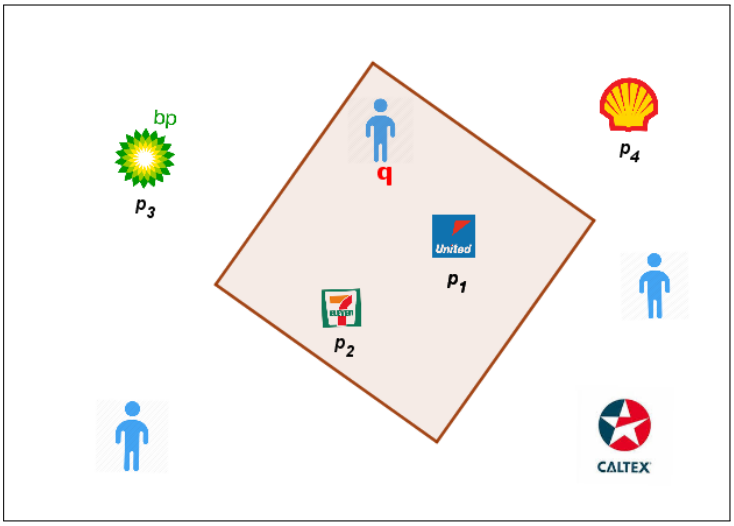


Figure 2.12: Safe region

The study conducted by [137] and [138] presents a rectangular-shaped safe zone for each continuous query in order to minimise the communication cost between clients and server in the continuous reverse nearest neighbour queries. This is done through the safe region which is intended to reduce the updating costs when a query is continuously moving inside the safe region. This is because the set of objects of interest does not change as long as the continuous query remains in this region. In this regard, many safe region methods have been proposed to achieve efficient evaluation by reducing the communication and updating costs. Some of these methods apply only time-based techniques [145], whereby objects need to report their query locations to the server after every t time units to obtain up-to-date information. Meanwhile others employ only distance-based techniques [146], whereby users need to report their query locations to the server after every d distance units. However, we note that these studies were designed to handle specific types of spatial queries, not all queries.

2.4 Clustering in Spatial Databases

Clustering is the process of grouping objects such that the objects in a group are similar (or related) to each other and different from (or unrelated to) objects in other groups. Clustering methods can be broadly classified into five groups: Partitioning, Density-based, Hierarchical, Grid-based and Model-based. Although cluster processing is not a new concept, there is not much published literature on clustering for spatial databases [147]. The examples of algorithms that consider spatial information are K-Means, K-medoids and Density-based clustering. K-Means clustering works involve arbitrarily choosing K point (centroids) which is K number of clusters.

Initially, K-Means randomly selects k number of points as centroids. Clustering points are then based on the distance to the centroids. If the points are closer to the centroids, then they belong to the core point. The updating of cluster centroids is based on the current assignments. The iteration cluster process continues until no further change occurs to the centroids. The K-Means approach has two limitations: it is sensitive, and it is efficient only when applied to small to medium-sized datasets [148].

CLARANS is another popular clustering algorithm and is intended to cluster a set of data based on randomized search [149]. Although this method is an improvement on the K-medoid method, it is still not efficient for medium-sized and large data sets. Nonetheless, none of the previous algorithms is able to cluster data in the most efficient way [150].

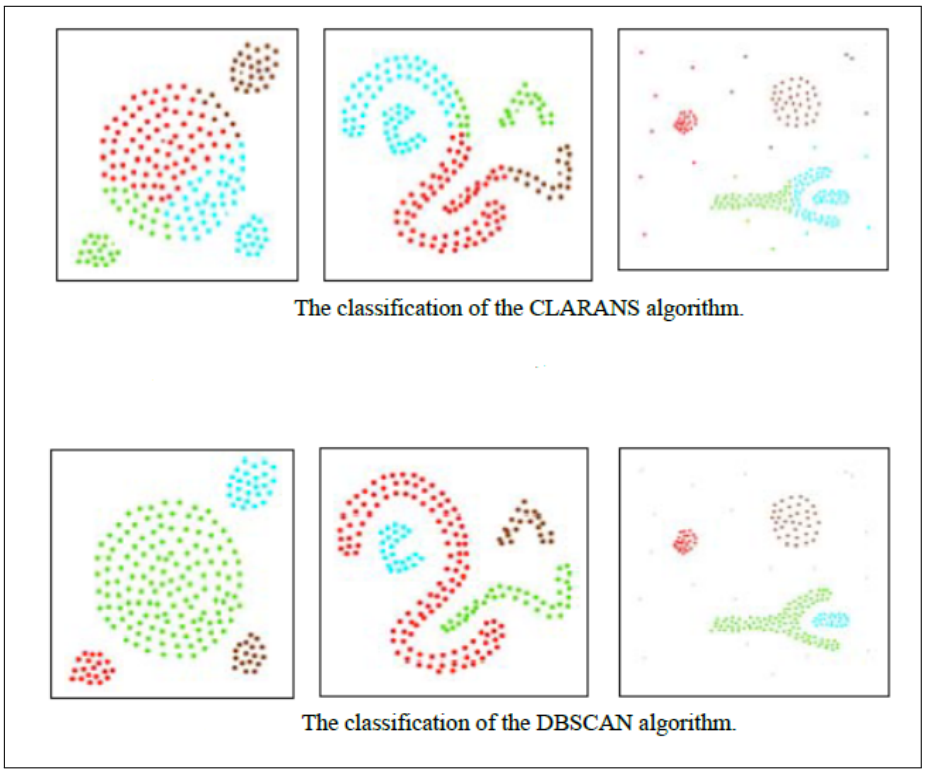


Figure 2.13: Comparisons between DBSCAN and CLARANS [4]

DBSCAN is a well-known density-based clustering algorithm which is involves partitioning points into dense regions [151]. [4] presented DBSCAN in a paper that received the highest impact award. It relies on a density-based notion of clusters. It uses two parameters Eps and MinPts, Eps values as the maximum radius of the group members. MinPts is the minimum number of group members in the Eps group of point. DBSCAN arbitrarily selects point q, from q and retrieves all points

density-reachable with respect to Eps and MinPts value. From density-reachable points p is selected; if p is a core point then the cluster is formed. However, if p is a border point , DBSCAN visits the next point in the database for investigation and the process continues until all of the points have been processed. DBSCAN is significantly more effective in discovering clusters of an arbitrary shape better than the well-known algorithm [150] as shown in 2.13. If the spatial index is used, the computational complexity of DBSCAN will be $O(nlong)$ instead of $O(n)$.

Several spatial queries have used the clusters based on the traditional queries, especially kNN queries [100] and [152], and a few are based on Reverse Nearest Neighbour (RNN) query [152] such as group nearest neighbour [9, 153] and Nearest Neighborhood (NNH) query [154]. Recently, several studies have investigated group nearest neighbour, which is a spatial query returning a group of points, proposed by [9] and further studied by others [155, 156] and [157]. [155] presents Group-KNN query, where the author handles multiple query objects instead of a single query such as the traditional range query and kNN query. In Group-KNN, given a set of query points, the query returns a single target object that has the minimum distance to all of the query points. Also, [158] studies the problem of finding group (k) points using a predefined measurement such as the distance between members.

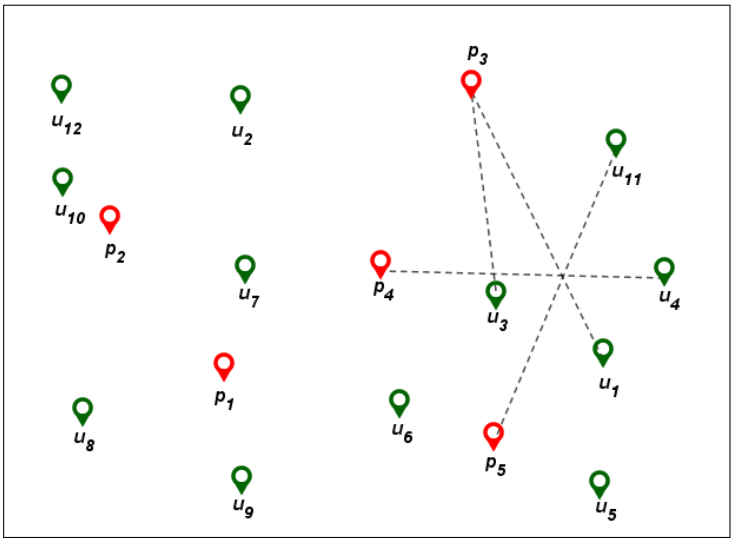


Figure 2.14: $GRNN(p4, p4, p5) = \{u1, u3, u4, u11\}$

Another interesting solution that has have been provided in the context of group points is Group Reverse kNN query, which was introduced by [159, 13]. Group Reverse kNN query is the method of finding points in p where k number is considered to be the query points in a set of F as their k nearest facility points. It is a special

modification of the RNN query. The Group Reverse kNN query is issued from a set of query points and the main goal is to find the objects that consider the set of query points as their nearest generator points. This type of query is illustrated in Figure 2.14. In this example, the query is issued from $p3, p4$ and $p5$, and the objective is to find all users u who consider these query points as their kNN. To answer the group reverse query, this kind of query returns $\{u1, u3, u4, u11\}$ users.

Another interesting query is known as Nearest Neighbourhood Search (NNH), which is an extended version of the NN query by proposing the use of nearest neighbourhood (NNH) query in spatial databases. Rather than returning the single closest neighbour, they present methods for finding the location of the nearest group of points, returning the centre of the circle that includes the group of points using the Nearest Enclosing Circle (NEC) geometric approach. [154] defines Nearest Neighbourhood query as follows: given a set p of points, a query point q, a positive number k, and a circle C of a given radius, the NNH query returns the nearest location (centre) of C such that the number of points covered (enclosed) by C is at least k. Let us consider Figure 2.15 which depicts NNH query and demonstrates the difference between NNH query and NN query. Suppose $k = 3$ and p_1 is closer to q than other points. Then the answer to the normal NN query is p_1. However, the NNH query will return c, which is the centre of the r-radius circle enclosing three points, p_2, p_4, and p_5, shown with a broken line. In other words, the NNH query presents a solution by clustering the nearest set of query objects that are close to each other.

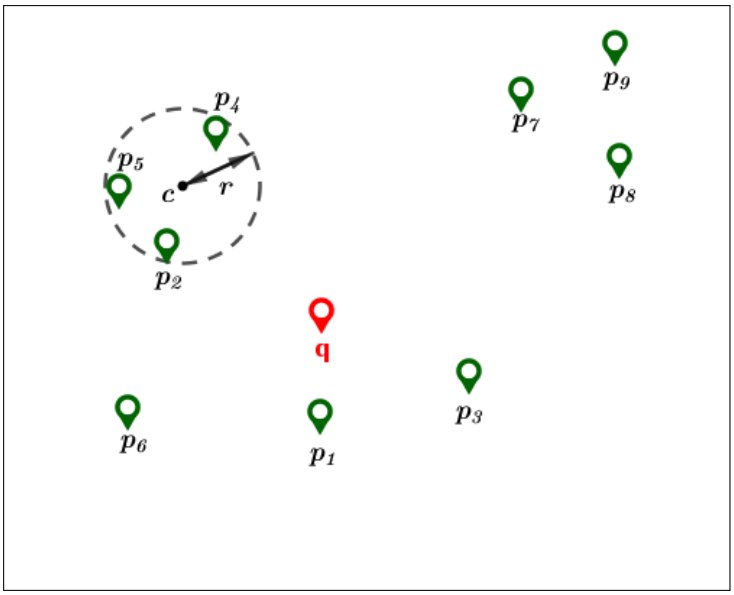


Figure 2.15: An example of the Nearest Neighborhood query (k = 3)

Although all these methods provide solutions that can be applied to cases involving Euclidean space, they are not appropriate for road networks which have particular non-Euclidean characteristics. The distance in a Euclidean space depends solely on the relative positions of two points, whereas the distance on a road network depends not only on the relative positions but on the section of road between the two points. To the best of our knowledge, little of the previous research has considered the notions of group or neighbourhood in spatial road networks.

2.5 Limitations

Based on the above discussion of published studies on spatial query processing in Euclidean and road networks environments, we found that several limitations need to be addressed. These are explained in more detail below.

2.5.1 Limitation of Existing Works in Traditional Spatial Queries

- **L1 :** Using a point-to-point-based calculation method for clustering the points makes the query processing very time-consuming.
- **L2 :** A region-based calculation method can be applied to a Reverse Nearest Neighbour RNN query; however, the candidate region members are widely dispersed members, and also it does not return all the influenced points for answering the query.
- **L3 :** The candidates of returning a group of points using Group Nearest Neighbour (GNN) query or nearest neighbourhood search would usually be far from the query. Furthermore, it does not consider the query point as the nearest of all facilities.
- **L4 :** The existing solutions and techniques used to define the group focus mainly on finding a group in a circular cluster using the Nearest Enclosing Circle (NEC) geometric approach; no existing work deals with a collection of several chained objects.

To address these issues, we cluster the points as a neighbourhood, which is a collection of a number of chained objects within a certain maximum distance between a pair of objects. The neighbourhood must ensure that these candidate points are not widely dispersed. Our proposed method uses a region-based method to improve the efficiency of our approach.

2.5.2 Limitation of Existing Works in Monitoring Continuous Spatial Queries

- **L1 :** The monitoring of continuous queries incurs very high communica- tion and computation costs.
- **L2 :** Existing analyses on continuous monitoring of spatial queries focus on reducing either computation or communication costs, while some studies concentrate on both. However, they pay little attention to the monitoring of spatial queries for a group version of points.
- **L3 :** Safe region studies were designed to handle only specific types of spatial queries.

In response to existing issues, we introduce the concept of safe region (SR) for continuous query point that keeps the reverse nearest neighbourhood result unchanged. Our proposed method aims to reduce the updating costs when a query is continuously moving in a safe region. This can continue as long as a query point in a safe region, query point (q), does not communicate to the server and the result of the point of interest does not change.

2.5.3 Limitation of Existing Works in Spatial Road Networks

- **L1 :** The previous proposed method involving Euclidean distance cannot be implemented directly in spatial road networks because there is a major difference between Euclidean space and road networks in terms of the distance metric used.
- **L2 :** To date, no study has considered the spatial road networks environment for computing Reverse Nearest Neighbourhood queries.

Hence, in this study, we develop solutions that can address the limitations of previous research on this topic. We propose a modified algorithm for reverse nearest neighbourhood query in road network settings.

Chapter 3

Reverse Nearest Neighbourhood (RNNH) Queries

3.1 Overview

The widespread use of location-aware services and technologies which retrieve or answer spatial queries has received much attention in today's society. An increasing number of popular applications, such as digital maps, make use of spatial databases and associated technologies. One of the most important branches of traditional spatial queries is the Reverse Nearest Neighbour (RNN) search. This search retrieves points of interest that consider the query facility as the nearest facility. Most of the existing works on spatial databases focus only on the retrieval of points of interest. Very little work has been done on grouping points of interest or on neighborhood retrieval. In this chapter, we introduce the concept of a group version of the Reverse Nearest Neighbour query called Reverse Nearest Neighborhood (RNNH) query. The RNNH query finds all possible reverse nearest neighborhoods where all the neighbourhoods consider the query facility as the nearest facility. We propose an efficient algorithm for processing snapshot RNNH queries by using an R-tree index. The proposed algorithm incrementally retrieves all reverse nearest neighbourhoods of the query facility. We have conducted exhaustive experiments on both real and synthetic datasets to demonstrate the superiority of the proposed algorithm.

3.2 Motivation

Spatial databases play an important role in modern software applications. Even in daily activities, we often rely on the aid of spatial databases such as digital maps when traveling from one place to another. Also, location-based services (LBS) are highly popular among some people as they provide location information in response

to queries [160]. Spatial databases also have applications in decision support systems, marketing strategies, and social networks.

The Reverse Nearest Neighbour (RNN) search is a fundamental issue that has been extensively studied in spatial databases for surveys [85, 12, 34, 161, 92, 89, 162, 24]. RNN finds every point of interest that considers the given query point q as the nearest point. Since q is close to such data points,it has a strong influence on these points. Hence, the set of points that constitutes the RNNs of a query point is called its influence set. RNN search queries can be exploited to gather information about the target customers of a facility or centre. If a facility or a centre is able to identify its target customers, then this can have a positive impact on their profitability. For example, location-based promotions can be offered to the target customers. However, because there may be a large number of customers in the vicinity of specific facilities or centres, it can be difficult to identify and reach individual customers (i.e., the point of interest). A Neighbourhood or a group of points of interest (GOI) is a concept which can assist in understanding the environment surrounding of the facility points and facilitating access to a cluster of customers. Therefore, the identification of neighbourhood can help with the development of GOI-oriented recommendation systems for spatial database applications.

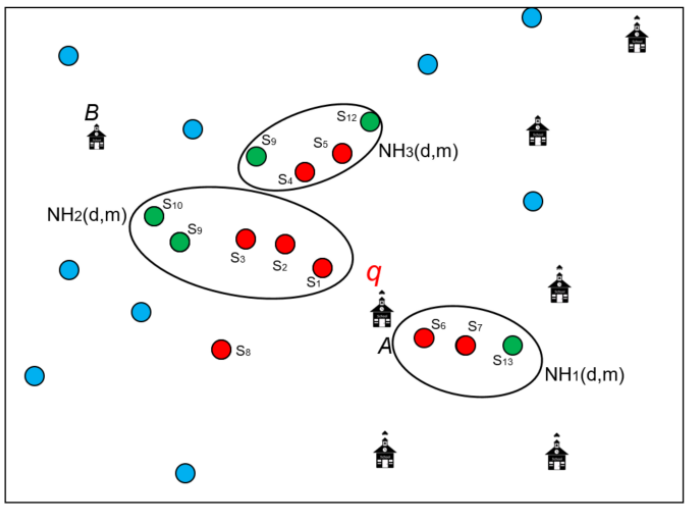


Figure 3.1: Example of a reverse nearest neighbourhood query

For example, consider a set of schools and a number of students ,as shown in Fig. 3.1 ,where each school attempts to attract students for enrollment. Some students can still choose school A due to the influence of their friends although school A could be further than other schools from the students' locations. This is probably because

a student receives a strong recommendation from his/her closest friend who goes to the same school. The Reverse Nearest Neighbour (RNN) identifies the students based on their proximity to the school. In this example, RNN query can identify only the students $\{s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8\}$ who are represented by the red points as the influence set of school A.

However, people consider various factors such as price, product quality, reputation and convenience factors in addition to distance from their location, and they typically choose a facility which best matches their preferences [15]. If we look at student s_9, can this student choose to go to school A instead of school B? If student s_9 looks up the most popular school in his area (around him), he can obtain school A because most students around him go to school A which is closer to them. School A can be one of the top choices for s_9 because the students who live around s_9 might strongly recommend school A to s_9. Therefore, we can say that student s_9 is close to other students who may prefer school A than the nearest school which is school B. In the neighbourhood concept, points are joined to one group based on their proximity. We name this Reverse Nearest Neighbourhood query (RNNH).

The example given in Fig. 3.1 shows the difference between the Reverse Nearest Neighbourhood query and the traditional spatial databases query. In this scenario, the output of the RNN will be the cluster of points s_1, s_2, s_3, s_4, s_5, s_6, s_7 and s_8. However, the neighbourhood will be the cluster of points s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}, s_{12} and s_{13}. All these students are clustered and returned in three groups NH_1, NH_2 and NH_3 based on their proximity to a neighbourhood. At first glance, it is obvious that a Reverse Nearest Neighbourhood query can be processed by clustering points using a sophisticated clustering algorithm, and finding all the clusters that are reverse nearest to the query point.

Nearest Neighbourhood vs. Reverse Nearest Neighbourhood Searches. The very first work on GOI-based data retrieval in spatial databases is nearest neighbourhood (NNH) retrieval [154]. The nearest neighbourhood (NNH) query finds m points closest to a given point and combines them into a neighbourhood. The authors model the nearest neighbourhood (NNH) by using the geometric Nearest Enclosing Circle (NEC) approach. NEC finds the centre of a circle nearest to q that has at least m points inside the circle. The resultant neighbourhood is a high density neighbourhood that has at least m points. The NEC method is capable of identifying one neighbourhood. However, RNNH is capable of identifying multiple neighbourhoods. Also, one of the unique features of RNNH is that in the instances where multiple neighbourhoods are identified, the neighbourhoods resemble chain

neighbours as opposed to density neighbours. The output from the NNH query is such that a scenario resembles density neighbours. As opposed to density neighbourhoods, chain neighbourhoods are more closely aligned with the spread of real-life physical structures like houses and strip shops. This means that the result of an RNNH query is more realistic.

DBSCAN vs. Reverse Nearest Neighbourhood Search. Another popular approach to clustering is the DBSCAN algorithm. [4] presented DBSCAN in a the research paper that received the highest impact paper award which is a testament to the popularity of the algorithm. The algorithm is based on partitioning points into neighbourhoods based on their density. It uses two parameters, Eps and MinPts, where, Eps is the maximum radius of the neighbourhood, and MinPts is the criterion for the minimum number of point's per neighbourhood. One of the shortcomings of the DBSCAN algorithm is that it considers only one kind of data point (e.g., user points), whereas, our approach considers multiple kinds of points (comprising users and facilities). The facilities can be one's own or associated with competition. Another limitation of DBSCAN is that it needs to access all points and continue processing until all points have been covered. Our algorithm utilises R-tree indexing and a pruning technique to prune the space, thus making our algorithm more efficient than DBSCAN in terms of CPU and IO costs.

Applications of Reverse Nearest Neighbourhood Search. This neighbourhood can play an important role in many applications of spatial databases. . Several examples of such are given below.

- **Decision Support System.** Many companies and business owners prefer to determine the influence of certain data points when they want to open a new store [100]. This is done to maximize the chances of the new store succeeding. Our proposed algorithm could be applied to identify the trade areas of a supermarket and its competitors if the geocoded locations of all supermarket customers is known. Instead of assigning each customer to a facility based on proximity, our algorithm will first identify clusters of customers and then assign those clusters to the respective facility. The supermarket can use such information to increase its market share and/or when planning new sites. In another example, clustering data near a specific hospital can give better insights into where accidents and emergencies mostly occur.

- **Mobile Social Network.** RNNH can be applied to social networks. By setting the radius parameter d at a certain value, the RNNH query can be utilized to extract a mobile community close to a user's location. For example, if there are tourist attractions such as a museums, it is assumed that each tourist will visit the closest museum. However, in reality, tourists usually look up the most popular museums the other people visit. A person might visit one museum which is not the one closest to him/her but is closer to other people. Therefore, choice of a museum can be influenced by a group of tourist attractions and places whose distances from each other are less than a particular threshold [163].

- **Targeted Marketing.** RNNH queries can be very helpful for discovering and identifying the neighborhood and area of influence. For example, if a restaurant, supermarket or gas station is a query point, it can be explored to discover the nearby neighborhoods prior to designing special promotion plans or deals. This requires that neighborhood members not be far from each other geographically. The query could minimize its travel cost associated with the promotion strategy, thereby possibly increase its revenue.

Contributions. The main contributions of this work are summarised below.

- We propose and define the concept of a group version of reverse nearest neighbour query, which we call reverse nearest neighbourhood (RNNH) query.
- We propose an efficient algorithm to process snapshot RNNH queries using R-tree data indexing.
- Our extensive experiments on real and synthetic datasets demonstrate the superiority of the proposed algorithm.

3.3 Notations and Definitions

Consider a set of facilities $F = \{f_1, f_2, ...f_k\}$, a query point $q \in F$, a set of points $P = \{p_1, p_2, ..p_n\}$ as the points of interest. Query point q where $q \in F$. Table 3.1 lists other notations used in this study.

Given a set of facilities F, a query facility $q \in F$ and a set of points P, the reverse nearest neighbour (RNN) query seeks points of interest $p \in P$ that consider the query point as the nearest of all facilities $f \in F$. Consider the example given in Figure 3.1, where s_1, s_2, s_3, s_4, s_5, s_6, s_7 and s_8 are the only points that consider q as their nearest facility. The RNN query result is denoted by $RNN(q)$.

Table 3.1: Notation and Definitions

Notation	Definition
q	the query point
$P = \{p_1, p_2, p_3, ...p_n\}$	a set of points, where n is a positive number
$F = \{f_1, f_2, f_3, ...f_k\}$	a set of facility points, where k is a positive number
p_i	a point p with ID i
$\overline{p_i p_j}$	a section of a line between two endpoints p_i and p_j
Z_q	the Voronoi diagram of the query q
Z_p	the points located inside the Voronoi diagram of the query q
$\|Z_q\|$	the number of points located inside the Voronoi diagram of the query q
$B_{q:f}$	a perpendicular bisector between points q and f
$H_{q:f}$	a half-plane defined by $B_{q:f}$ containing the query q
$NH(d, m)$	a neighbourhood consisting of a group of at least m points
d	a Euclidean distance parameter between two points
m	a parameter of minimum number of points inside a neighbourhood
$\|NH(d, m)\|$	the number of points located inside the neighbourhood $NH(d, m)$
f_{p_i}	the nearest facility to p_i
$dist(p_i, p_j)$	the minimum Euclidean distance between points p_i and p_j
$d_H(p_i, NH)$	the minimum Euclidean distance between a point (p_i) and a neighbourhood $NH(d, m)$

Definition 3.1 ***(Boolean Range Query)*** *Given a set of facilitates F and a distance constraint d, a boolean range query is denoted by $boolRangeQuery(p_i, d, F)$, where $p_i \in P$, returns TRUE if there is any facility $f \in F$ such that $dist(p_i, f) \leq d$, otherwise returns FALSE.*

For example, consider the dataset given in Fig. 3.2. The Boolean range query for s_9 with distance constraint $d = 5$ returns TRUE as $dist(s_9, B) \leq 5$ (as shown in Fig. 3.2), but with distance constraint $d = 3$ it returns FALSE.

Definition 3.2 ***(Neighbourhood)*** *Given two parameters m and d, a neighbourhood refers to a group of at least m member points where the distance between a member point p_i and the nearest member point p_j in the group does not exceed d, i.e., $d(p_i, p_j) \leq d$. The neighbourhood is denoted by $NH(d, m)$.*

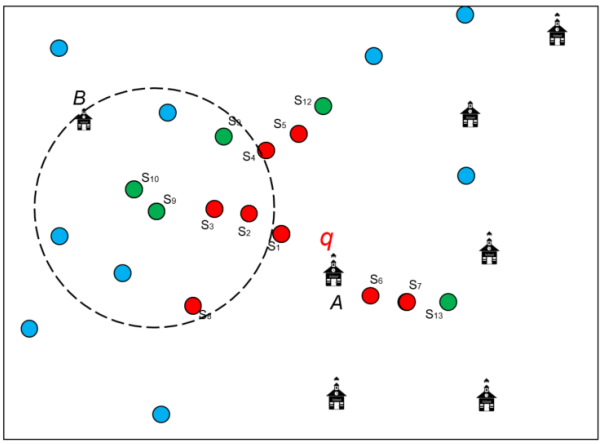


Figure 3.2: Example of a boolean range query

The neighbourhood $NH(d,m) = \{p_1, p_2.., p_m\} \in P$ refers to a neighbourhood with Cartesian coordinates $\{(x_{11}, x_{12}), .., (x_{m1}, x_{m2})\}$ in the spatial data space for P, where $1 < m \leq n$ and $p_i \neq p_j$, $\forall i, j \in \{1, 2, ..., m\}$. The distance parameter d denotes the maximum distance between a member point of $NH(d,m)$ and its nearest member point in $NH(d,m)$ and the cardinality constraint m refers to the minimum number of neighbourhood members. For notational simplicity, here we use NH instead of $NH(d,m)$.

Figure 3.3 gives an example of neighbourhood queries in spatial databases. Assume that we are looking for only those neighbourhoods that satisfy the constraints $m = 3$ and $d = 3$. Then, $NH_1 = \{p_1, p_2, p_3, p_4\}$ and $NH_2 = \{p_6, p_7, p_8, p_9\}$ are neighbourhoods that contain a group of points that has at least three members with each group member having a threshold distance ($d = 3$) to the nearest group member. The point p_{15} is not part of NH_1 because the distance between p_{15} and its closest point p_1 in NH_1 is > 3. The group of points $\{p_{10}, p_{11}\}$ is not a neighbourhood as the number of neighbourhood members is only two, which does not satisfy the cardinality constraint $m \geq 3$.

Definition 3.3 *(**Neighbourhood Distance**) Given a neighbourhood $NH(d,m)$ and a point p_i, where p_i being a point of interest or facility, the neighbourhood distance refers to the distance between the point p_i and the closest point in the neighbourhood $NH(d,m)$. This distance is denoted by $d_H(p_i, NH)$ and can be formalized as:*

$$d_H(p_i, NH) = min\{dist(p_i, p_j) | p_j \in NH(d,m)\} \tag{3.1}$$

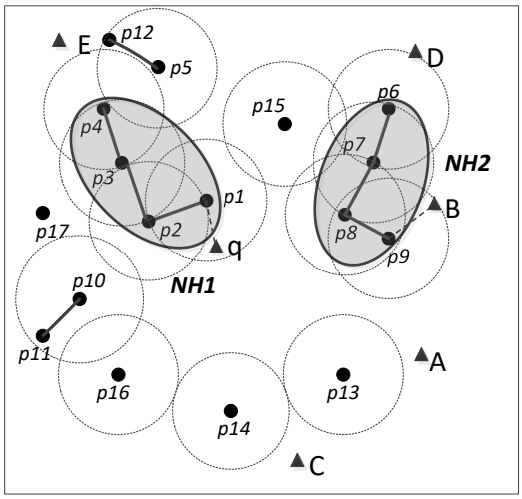


Figure 3.3: Example of neighbourhood queries $NH(d = 3, m = 3)$

Figure 3.4 illustrates an instance of neighbourhood distance, where $NH = \{p_1, p_2, p_3, p_4\}$ is a neighbourhood. Now, $d_H(p_{10}, NH)$ is the distance between p_{10} and the nearest member point of NH to p_{10}, i.e. the distance between p_{10} and p_2 ($dist(p_{10}, p_2)$). Similarly, $d_H(E, NH)$ is the distance between E and the nearest member point of NH to E, i.e, $dist(E, p_4)$. Again, $d_H(q, NH)$ is the distance between q and the nearest point of NH to q, which is $dist(q, p_1)$.

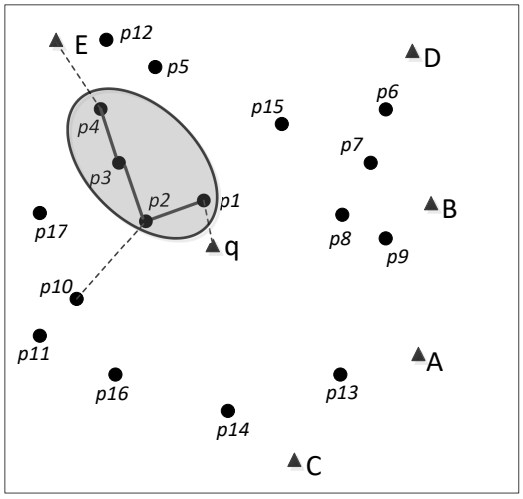


Figure 3.4: Neighbourhood distance

Definition 3.4 *(**NNH Query**) Given a a query facility q, a set of points P, two constraints d and m, and neighbourhoods $\mathcal{S} = \{NH_1, NH_2, ..., NH_{n'}\}$, where $NH_i \subseteq P$, $\forall i \in \{1, 2, ...n'\}$, a nearest neighbourhood (NNH) query for q returns the neighbourhood $NH_i \in \mathcal{S}$ such that $d_h(q, NH_i) \leq d_h(q, NH_j)$, $\forall NH_j \in \mathcal{S} \setminus NH_i$. The NNH query is denoted by $NNH(q, d, m, P)$.*

Consider the neighbourhood example illustrated in Figure 3.3. Here, NH_1 is the neighbourhood, which is the nearest neighbourhood for q for parameters $d = 3$ and $m = 3$. It should be noted that there could be a tie in the NNH query result. In this case, we can apply random selection to break the tie. As long as there is a neighbourhood in the dataset P for the given parameters d and m, the NNH query result can never be empty.

Definition 3.5 *(**Reverse Nearest Neighbourhood (RNNH) Query**) Given a set of facilities F, a query facility $q \in F$, a set of points P, two constraints d and m, and neighbourhoods $\{NH_1, NH_2, ..., NH_{n'}\}$, a reverse nearest neighbourhood (RNNH) query for q returns all neighbourhoods $\{NH_i\}$ such that: (i) the distance of a point $p_j \in NH_i$ to its nearest neighbour point $p_k \in NH_i$ is less than or equal to d value, i.e., $dist(p_j, p_k) \leq d$; (ii) $\forall p_j \in NH_i$, $d_H(p_j, NH_i) \leq dist(p_j, f_{p_j})$, where f_{p_j} is the nearest facility of p_j in F; (iii) $|NH_i| \geq m$; and (iv) NH_i finds the query facility q as their nearest facility among all facilities in F, i.e., $d_H(q, NH_i) \leq d_H(f, NH_i)$, $\forall f \in F \setminus q$. The RNNH query is denoted by $RNNH(q, d, m, P, F)$.*

In simplest terms, the RNNH query retrieves neighbourhoods $\{NH_i\}$ that consider the query point as the nearest of all the other facilities. Consider the facilities $\{A, B, ..., E\}$ and the points $\{p_1, p_2, ..., p_{16}\}$ in the space as depicted in Figure 3.3. Here, the result of RNNH query for the query facility q is NH_1 for parameters $d = 3$ and $m = 3$ as $d_H(q, NH_1) \leq d_H(f, NH_1)$, $\forall f \in \{A, B, ..., E\}$. The neighbourhood NH_2 is not a reverse nearest neighbourhood of q as $d_H(B, NH_2) < d_H(q, NH_2)$. It should be noted that there could be zero or more reverse nearest neighbourhoods of a query facility q in a given dataset $P \cup F$ as opposed to the nearest neighbourhood. Unlike an NNH query, with a RNNH query, we also need to deal with the competitor facilities.

Definition 3.6 *(**Influence Zone**) Given a set of facilities F, a query facility $q \in F$ and a set of Points P, an influence zone of q, denoted by $Z(q)$, is an area*

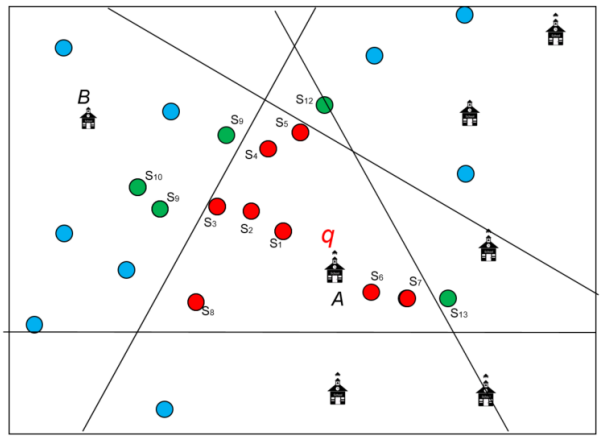


Figure 3.5: Influence zone of q

which is a convex polygon and every point p_i in this area considers the query facility q as the nearest facility, i.e., $\forall u \in Z(q)$, $dist(p_i, q) \leq dist(p_i, f)$, $\forall f \in F \setminus q$ [24].

Consider the dataset given in Fig. 3.1. The influence zone of the query facility q is shown in Fig. 3.5 which contains red dots.

3.4 Proposed Framework

3.4.1 Possible Solutions

Obviously, a simple approach for the RNNH query is to use the baseline algorithm which computes the neighbourhoods by starting with any point $p \in P$. From this selected point, the neighbourhood NH_i is established as a starting point. This NH_i expands by doing a range query for each point in NH_i. Hence, each neighbourhood is expanded by repeatedly executing the range query for each user $p_i \in NH_i$ until $|NH_i| = m$ and follows the distance constraint d and conditions given in Definition 5.4. For each neighbourhood NH_i is $d_H(p_i, NH_i) \leq d_H(p_i, f)$ and $d_H(q, NH_i) \leq d_H(f, NH_1)$, $\forall u_i \in NH_i$ and $\forall f \in F \setminus q$. If the p is not satisfied in Definition 5.4, then add p to S (i.e., this is not a valid neighbourhood of q). This step will require only $\mathcal{O}(m\ log\ n)$ which is much faster than the combination dataset. Next, we repeatedly start new iterations for each point p that has not been covered in any previous valid neighbourhood of q and $p \notin S$ by forming neighbourhoods and expanding it by doing a range query. There is a possibility of some users obtaining overlapping neighbourhoods in some cases. Once all neighbourhoods have been formed, then we start to maximize a neighbourhood by including as many additional users as possible by executing a range query as per Definition 5.3 and Definition 5.4.

Unfortunately, the baseline algorithm that we described is computationally expensive. In a practical situation, the baseline algorithm is too large to be preprocessed in an index structure or to be processed on the fly. One possible solution for this problem is invoking incremental nearest neighbour to form RNNH (KnnRNNH). Given a query point q, retrieve the first nearest neighbour point u_i. If the point satisfies the distance constraint d and conditions given in Definition 5.4: (a) $d_H(p_i, NH_i) \leq dist(p_i, f)$ and (b) $d_H(q, NH_i) \leq\ d_H(f, NH_i)$, $\forall p_i \in NH_i$ and $\forall f \in F \setminus q$, then form a neighbourhood. From p_i invoke the kNN query. Mark as visited all the points that have been retrieved. Any of the retrieved points that satisfy the same constraints, form a neighbourhood. When the distance between a retrieved point and p_i is greater than d, then stop invoking the kNN query for p_i. Repeat these processes for each element in the neighbourhood until all users satisfy the distance constraint d and the conditions given in Definition 5.4. The formation of the first neighbourhood is terminated when the retrieved users of a kNN query for each element of the neighbourhood is greater than the d value. Then start forming the next neighbourhood by invoking the second nearest user from the query that is not included in first neighbourhood. The RNNH stops forming neighbourhoods when the nearest neighbour visited from the query point is greater than d value. We use this approach (KnnRNNH) to compare it with our algorithm. In the baseline algorithm, the probability of getting a reverse nearest neighbourhood (RNNH) is low because the selection of the initial point of interest is random.

In our algorithm, we use the influence zone to skip the process of iteratively selecting the points. The initial points of interest are selected within the influence zone. The probability of a group of points being selected from within the influence zone, and these selected points of interest belonging to the same RNNH is very high. This approach makes our algorithm much faster than the baseline algorithm. Our propsed RNNH algorithm utilises R-tree indexing and pruning techniques to prune the space. In terms of computational complexity, the R-tree indexing of RNNH saves time otherwise spent in creating of the influence zone. Additionally, the use of pruning in RNNH makes this algorithm less complex than any other clustering algorithm such as DBSCAN as it does not have to go through the entire space.

3.4.2 Proposed Algorithm

In this chapter, we have extended the RNN query by considering a group version of this query. We have named the resultant query, RNNH. The goal of RNNH is to find the location of the nearest group of points based on preferences specified in the query.

RNNH returns a number of neighbourhoods that have a group of points which is different from returning individual points of interest as most other traditional spatial queries do.

Based on the definition of 5.4, the aim is to identify groups of points, often represented as a cluster of points, which are generally closer to q. In other words, the clusters identified will be those in which the points are close not only to each other, but also to q. RNNH retrieves all the groups of points that only consider q as the nearest facility.

To address the challenge of processing a RNNH query, we have proposed an R-tree based algorithm to obtain better performance. Initially, the InfoZone Z is created based on the RNN [24]. It is generated based on the bisector of the dataset of F. Consequently, any points in $H_{q:f}$ can be part of the RNNH answer. The algorithm for retrieving the points inside Z has been discussed in [24]. Every point on the boundary of Z, it is $u \in Z$. Hence, if the Z of a query q is empty ($Z = \emptyset$), then there is no reverse nearest neighbourhood for q.

Lemma 1 *If the influence set of a query location q, denoted by Z_q, is empty, then there is no reverse nearest neighborhood for q.*

Proof 1 *Given facilities F and points P such as $q \in F$ and $\forall p \in P$. The influence zone is created by placing the bisectors between the q and other facilities. There is no RNNH result, if there are no points of interest within Z, which means $Z_q = \emptyset$. In this case, all the points could not consider the q as the nearest facility. Hence, the nearest point p to q is in the $H_{f:q}$ (i.e., $dist(p, f) < dist(p, q)$). So, $\forall NH(d, m)$ $d_H(q, NH(d, m)) \not\leq d_H(f, NH(d, m)$. Therefore, in this case, all nearest points to q do not consider q as the nearest facility. This is because the nearest point of interest p to q is $dist(p, f) < dist(p, q)$. Thus, $\forall p \in P$, $dist(p, q) \not\leq dist(p, f)$.*

Once, Z_q is created and a point is retrieved, RNNH starts processing the first neighbourhood by picking up the first point from inside Z_q. RNNH starts at the point close to q to generate the first neighbourhood taking into account the minimum number neighbourhood members m, and the distance between a point to the nearest point d inside a neighbourhood. [1]

[1] It should be noted that there could be more than one candidate in Z_p that has the same distance to q. We randomly pick one of them to break the tie.

Algorithm 1 Computing Reverse Nearest neighbourhood

Input: P: set of points, F: set of facilities, d: distance constraint, m: minimum neighbourhood members and q: $q \in F$

Output: $RNNH\text{-}RN(d, m, q)$

$Z_q \leftarrow$ Calculate Influence zone of q

$InfZonePoint \leftarrow$ get all points in Z_q

$h \leftarrow$ store with key $mindist(InfZonePoint, q))$

while *h is not empty* **do**
  de-heap an entry p_i
  if $p_i \notin RNNHs(d, m, q)$ **then**
    mark p_i as visited
    initialize $NH_i \leftarrow \{p_i\}$
    $dist \leftarrow d_H(q, NH_i)$
    validationList $\leftarrow rangeQuery(p_i, d, P)$
    while *validationList is not empty* **do**
      de-heap an entry c_i
      if $c_i \in NVD_p$ **then**
        mark c_i as visited
        $NH_i \leftarrow append(c_i)$
      end
      else if $isValid(c_i, dist, q, NH_i)$ **then**
        $NH_1 \leftarrow append(c)$
        $L \leftarrow rangeQuery(c, d, U)$
        validationList $\leftarrow append(L)$
    end
    if $|NH_i| \geq m$ **then**
      $RNNH(d, m, q, P, F) \leftarrow append(NH_i)$;
    end
  end
end

Algorithm 1 is the algorithm used for processing the RNNH query based on the incremental retrieval of nearest neighbours resulting from the initial query. It uses different techniques to maintain sets of retrieved nearest points from q, such as the use of multi-heap technique to track the result set. Whenever the Z is created, the algorithm puts the points, which are inside the Z, in the min heap in order of the minimum distance to q (line 6). It then starts processing from the nearest point to q. Line 9 checks each point to determine wheather or not it belongs to a previous neighbourhood; if it does, then it is ignored. Otherwise, a new neighbourhood is initialized and puts this point in this neighbourhood (line 10). Line 11 is the function

for obtaining the $d_H(q, NH)$ which is the distance between q point to NH_i. In line 12, a range query is executed for the entry p_i.

The result of the range query is inserted in the heap (Validationlist) with the key minimum distance to the query. From the validation list, points are iteratively retrieved from the front of the heap(Line 14). Any point which has already been processed is ignored. Otherwise, it is checked to establish whether or not it is eligible of the NH. For each point in the validation list (Line 18) is checked before being added to the NH. This process ensures that the point is not closer to any another facility than to q as per the definition in 5.4 and 5.3. If the point is valid. it is added to NH_i; after that the validationlist is updated using a range query.

Pre-computation. We pre-computed the nearest facility to each user with the distance to improve the efficiency of the algorithm. The RNNH algorithm needs to do a validation for the set U to validate the users as per the definition in 5.4. The pre-compute process helps with the validation by ensuring that each user p_i is $d_H(p_i, NH_1) \leq dist(p_i, f)$. It improves the performance of the algorithm in terms of the IO cost with regards to validating the user.

Lemma 2 *For any retrieved point e where $e \in Z_q$. Then, an entry e is a valid entry if the distance between e and NH_1 is $\leq d$.*

Proof 2 *To prove that e is a valid entry, given the set of facilities $f \in F$, $q \in F$ and set of points $p \in P$. e is inside of Z. This means e lies on $H_{q:f}$. As per the definition of the bisector [162], any points in $H_{q:f}$ considers q is the nearest facility among any other facilities which means that $dist(p_i, q) < dist(p_i, f_i)$ as per the definition 5.4. Hence, we do not need to check the validation of points in Z_q , because any point that is there has already been validated as per definition 5.4 and added to NH_i.*

This lemma shows how we can address the points if the entry of the validation list is inside Z; such a point does not require extra processing, therby, improving the performance of the algorithm.

Influence Zone Optimisation. For any point inside of Z, there is no need to check the validation process for these points as per lemma 10.

Once the neighbourhood NH_i has been completely processed, check whether NH_i is satisfied with the m, then add it to RNNH. Otherwise, start another point from Z_q that has not been processed.

The function of isValid is precomputed function. It is how to make sure whether the point is valid based on the two constraints given in the definition 5.4. The first constraint is the distance between the neighbourhood and query, and the second constraint is the distance between the point and the nearest point in the neighbourhood. The grater of the two constraint values is used as a value of checking. If that is true then add this point to NH_i. The $isValid$ function guarantees that the NH_i is not much closer to any competitor facilities. The idea here is to make sure that no point is closer in distance compared to the competitor facility.

MaxDist Optimisation. Here, instead of doing a double check for the two constraints, we take the maximum distance between $d_H(q, NH)$ and the distance between the point to the nearest point inside NH. This improves the performance of this algorithm. We need to make sure that there is no competitor facility closer than q point or current neighbourhood. We consider the maximum distance between $d_H(q, NH)$ and $d_H(p_i, NH)$ and do one check instead of two validations. Assume that the nearest facility of each point $p \in P$, denoted by $NF(p)$, is precomputed. Then, line 20 of Algorithm 1 can be optimized as follows: if $maxDist < dist(c, NF(c))$, then return *false*. As a result of this approach, our algorithm (ORNNH) is optimized compared to the non optimized algorithm and it is cheaper, as demonstrated by the results reported in the experimental section.

3.5 Experimental Results

3.5.1 Experimental Setup

In this section, we provide the detailed experimental evaluations of our proposed methods in comparison with other competitors.

Competitor. All algorithms were implemented in C++ and experiments were run on Intel Core I 5 2.3GHz PC with 8GB memory running on Ubuntu Linux. To the best of our knowledge, currently there is no algorithm available for dealing with RNNH queries. We have considered a naive algorithm (KnnRNNH) query and have made reasonable efforts to devise a significantly improved version of KnnRNNH, as explained below.

Given a query point q, retrieve nearest neighbor user p from q such that the distance between q and u is less than the distance between u and the nearest competitor facility. A kNN query is invoked to retrieve any user p_1 where the distance between p_1

Table 3.2: Experiment Dataset

Datasets	Description	#Points
$SYN1$	Synthetic Normal Distribution	65 K
$SYN2$	Synthetic Normal Distribution	130 K
$SYN3$	Synthetic Normal Distribution	650 K
$SYN4$	Synthetic Normal Distribution	1.3 M
NA	North America	175 K
LA	Los Angeles	2.6 M

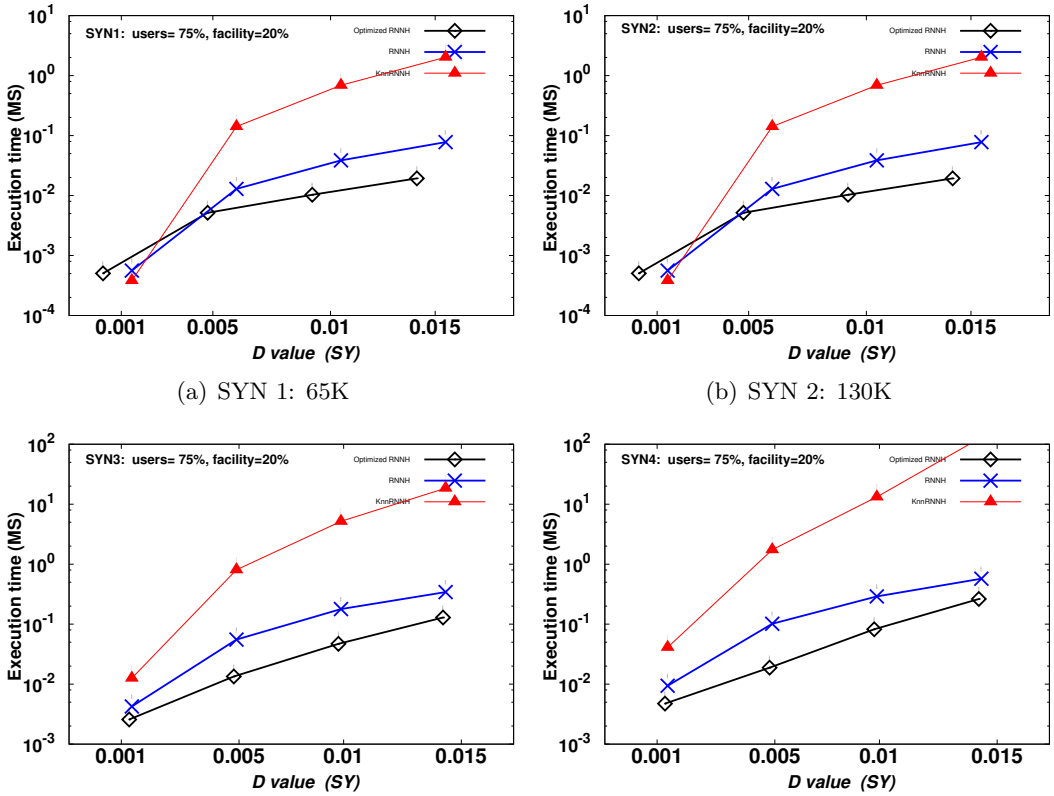
and p is less than d. Meanwhile, the distance between p_1 and the nearest competitor facility must be greater than $d_H(q, NH)$. After that, for each newly retrieved user, a kNN query is conducted to find the users that satisfy the same constraints. All retrieved users are marked as visited and form part of as a neighborhood. When no more users can be added, then this neighbourhood is returned as the result. For unvisited users, the nearest user to q is retrieved and then the process is repeated to find the neighborhood. The process terminates when all users are marked as visited or the distance of the nearest user to q is greater than the distance to a competitor.

Datasets. We generated several synthetic data sets containing 70,000 to 125,000, 700,000 and 1.3 million points following normal distributions. Table 1 shows the details of the synthetic data sets that were used for the experiments. The number of the points shown in Table 1 for each data set is randomly divided into two sets of 25 percent and 75 percent, one corresponding to the facility points and the other to the user points. In addition to the synthetic data sets, we also conducted experiments on two real data sets. We used real data sets containing 175,812 points from all across North America (called NA data set hereafter), 2.6 million points from the city of Los Angeles (LA) in North America. The page size of each R*-Tree [164] was set to 4,096 Bytes. We randomly selected 50 points from the facility data set and treated them as query points. The default value of d is 0.005 and 20% is facility percentage and 75% is the percentage of users in the whole data set. We considered two algorithms in our experiments: RNNH, and KnnRNNH. As we explained in the previous section, we used a optimization technique in our RNNH solution. In this experiments section, we compare three algorithms: the Optimized RNNH algorithm, the RNNH algorithm, and the KnnRNNH algorithm.

3.5.2 Evaluating Performance on Synthetic datasets

In this section we evaluate the performance of our algorithms for snapshot RNNH queries. All three algorithms need to traverse facilities and users utilizing R*-tree

indexing. We conducted experiments by varying three parameters: d value (0.001, 0.005, 0.01 and 0.015), percentage of facilities (0.1, 0.15, 0.2 and 0.25) and percentage of users (0.15, 0.35, 0.55 and 0.75). The results from the experiments are summarized in Figure 3.6 to 3.8.



(a) SYN 1: 65K

(b) SYN 2: 130K

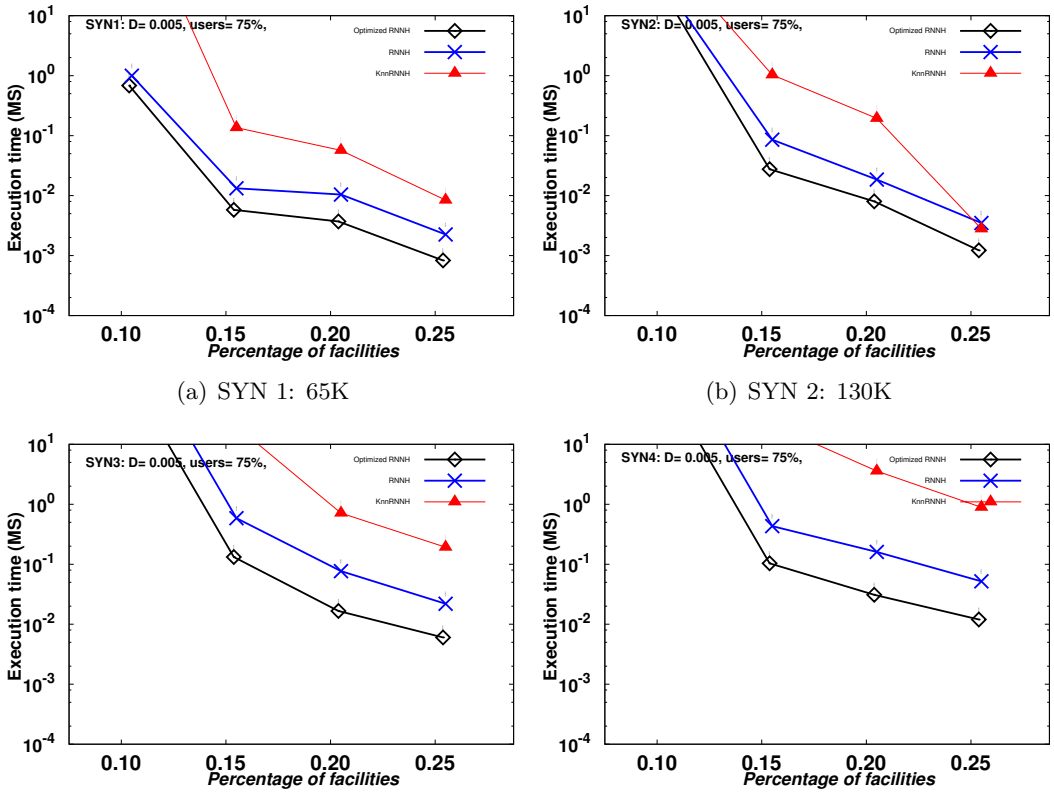
(c) SYN 3: 650K

(d) SYN 4: 1.3M

Figure 3.6: Effect of varying the number of d

Varying d: We evaluated the performance of RNNH query on both real-world and synthetic datasets. We varied d in increments of 0.001, 0.005, 0.01 and 0.015. The processing time in 4 synthetic data sets are shown in Figure 3.6. Several conclusions were drawn. Firstly, the average processing time for an RNNH query achieved by the optimization RNNH algorithm is the shortest of the three algorithms. With the increasing value of d, the difference becomes much larger. As we can see from Figure 3.6(a), the optimization RNNH algorithm and KnnRNNH algorithm perform almost the same when the value of d is small. However, when the value of d increases, the RNNH algorithm is superior to the RNNH algorithm. This is because with small values of d the resultant neighbourhood is not big, which means that it is not able to process more points.

However, when the values of d increase, the differences between the RNNH algorithm and the knnRNNH algorithm in terms of performance increase, where the much faster RNNH outperforms KnnRNNh algorithm. As expected, the running time of each algorithm increases with an increase in the value of d in all the data sets. Secondly, in Figure 3.6(c), we can see the difference between RNNH and knnRNNH algorithms, even with a small d value. As mentioned before, the reason behind that RNNH algorithm outperform otheralgorithms is that it works better with large data sets and KnnRNNh starts processing many points compared with the RNNH algorithm, which starts processing only the point close to q which is inside the Z area. This is also demonstrated in Figure 3.6(d) with 1.3 million points.



(a) SYN 1: 65K

(b) SYN 2: 130K

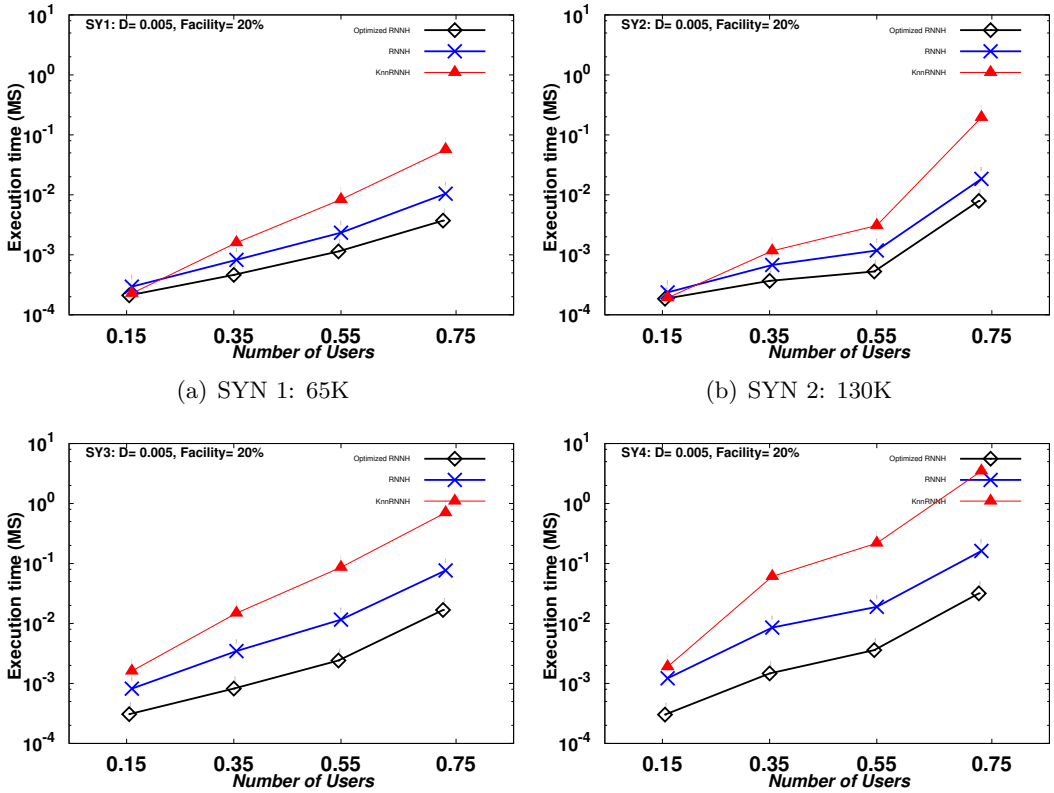
(c) SYN 3: 650K

(d) SYN 4: 1.3M

Figure 3.7: Effect of varying the number of facilities

Varying percentage of facilities: In Figure 3.7, we varied the percentages of facilities in 10%, 15%, 20% and 25% of the total of dataset. As shown in Figure 3.7(a), SYN1 is a small data set, however the RNNH is superior in the context of varying the number of facilities in all the patterns of percentage. Secondly, it is very clear as we have seen in Figure 3.7(d) that the CPU cost declines with an increase in the number of facilities. The overall trend is that the smaller the percentage of

facilities, the longer is the execution time. This is because the overall number of processing points increases as the number of facilities decreases.



(a) SYN 1: 65K

(b) SYN 2: 130K

(c) SYN 3: 650K

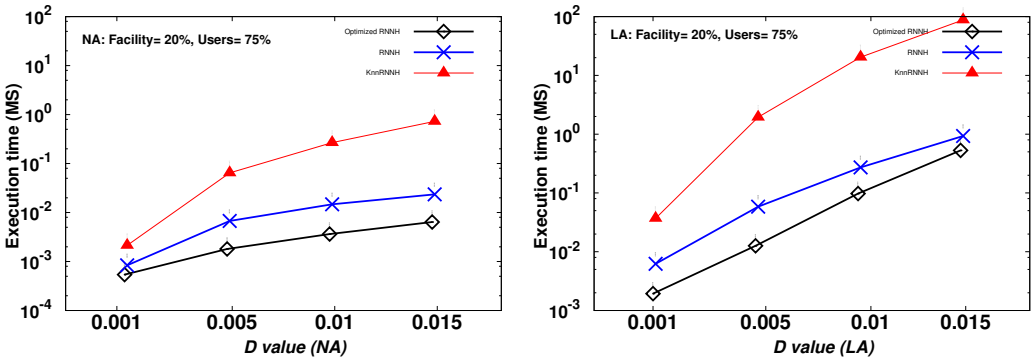
(d) SYN 4: 1.3M

Figure 3.8: Effect of varying the number of users

Varying the percentage of users : We evaluated the performance by varying the number of users and observing the effect on the CPU cost. This is shown in Figure 3.8. The percentages of users are 15%, 35%, 55% and 75% of the data set. We can see in Figure 3.8 that the processing time increases as the number of users increases. This is because there is a correlation between the number of users and the execution time: when the number of users increases, the execution time increases. Also, it becomes worse for the knnRNNH algorithm because it is processing many more points compared with RNNH algorithm which only processes points that are closed to q.

Secondly, in Figure 3.8(a), we can see that for a small percentage of users, all three algorithms performed equally well. However, the CPU cost increases with an increase in the number of users. This is because when the total number of points, facilities and users are similar, the size of the neighbourhood will be small and this

decreases the processing time. Finally, for the KnnRNNH algorithm the rise is very dramatic compared to our algorithm.



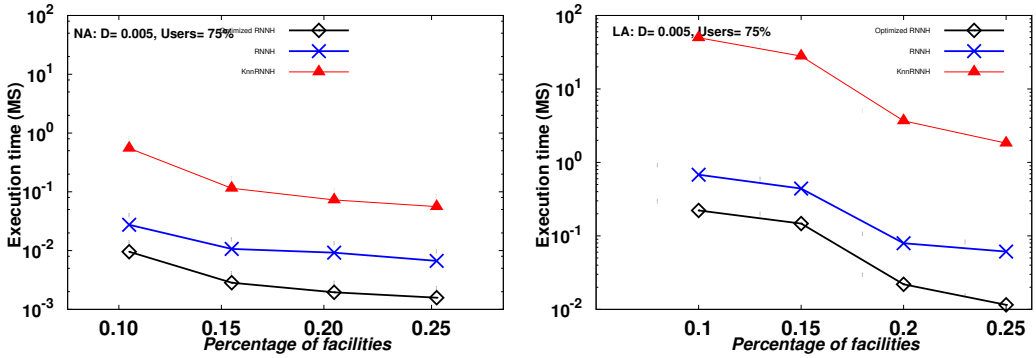
(a) NA : 175K (b) LA : 2.6M

Figure 3.9: Effect of varying the number of d

3.5.3 Evaluating Performance on Real Datasets

Figures 3.9 to 3.11 show the effect of varying the number of d, percentage of facilities and users on real datasets (NA and LA). Note that the NA dataset is much smaller than the LA dataset (175k vs 2.6 million, respectively). In terms of the effect of varying the number of d, as expected, the general execution time for all three algorithms was less for the NA data set than for the LA data set as shown in Figure 3.9. Secondly, for both data sets, optimized RNNH had the shortest execution time, and the differences between the execution times for KnnRNNH, RNNH and optimized RNNH are much greatest for the LA data set. This indicates that as the data sets become larger, RNNH and optimized RNNH are more efficient than KnnRNNH algorithm.

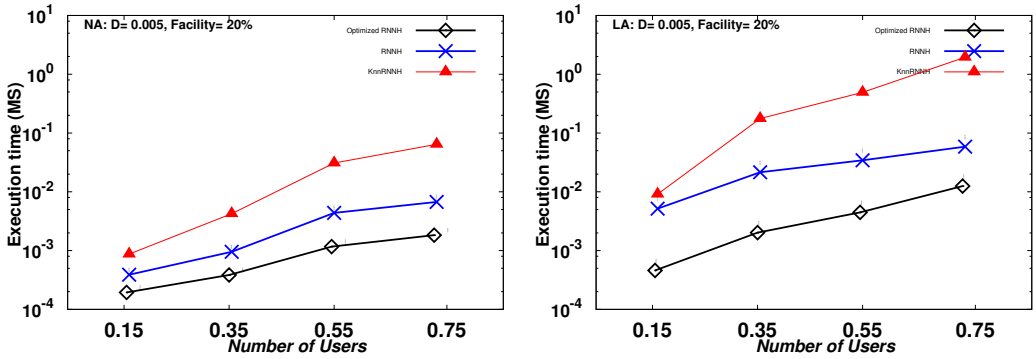
Figure 3.10 shows the effect of varying the number of facilities. The execution time for all three algorithms was less for the NA data set than for the LA data set. This because there is a correlation between the number of points and the execution time. Secondly, for both data sets, the optimized RNNH had the least execution time, this was followed by RNNH, and finally, KnnRNNH. However, the difference between the execution times for KnnRNNH, and optimized RNNH and RNNH is much greater for the LA data set. Figure 3.11 shows the effect of varying the number of users, the general execution time for all three algorithms was less for the NA data set compared to the LA data set. For both data sets, the optimized RNNH had the least execution time, this was followed by RNNH, and finally, the KnnRNNH algorithm.



(a) NA : 175K (b) LA : 2.6M

Figure 3.10: Effect of varying the number of facilities

However, the difference between the execution times for KnnRNNH, and RNNH and optimized RNNH is much greater for the LA data set. When comparing the RNNH and optimized RNNH, the LA data set had a much larger difference between the efficiency of optimized RNNH and RNNH compared to the NA dataset but the difference decreased as the number of users increased. This indicates that as the data sets become larger, the optimized RNNH is the most efficient, followed closely by RNNH , and finally, KnnRNNH.



(a) NA : 175K (b) LA : 2.6

Figure 3.11: Effect of varying the number of users

3.6 Conclusion

In this chapter, we have introduced the concept of a group version of Reverse Nearest Neighbour called Reverse Nearest Neighbourhood (RNNH). We proposed RNNH

algorithms using R-tree indexing. This algorithm is used for the incremental retrieval of reverse nearest neighbours. It finds all possible reverse nearest neighbourhoods instead of only the nearest reverse points. All the neighbourhoods consider the query as the nearest facility. We have conducted exhaustive experiments on real and synthetic datasets to demonstrate the results of these proposed methods. The results found that the proposed algorithm minimizes the total running time to process and find all the nearest reverse groups.

Chapter 4

Continuous Reverse Nearest Neighbourhood Queries

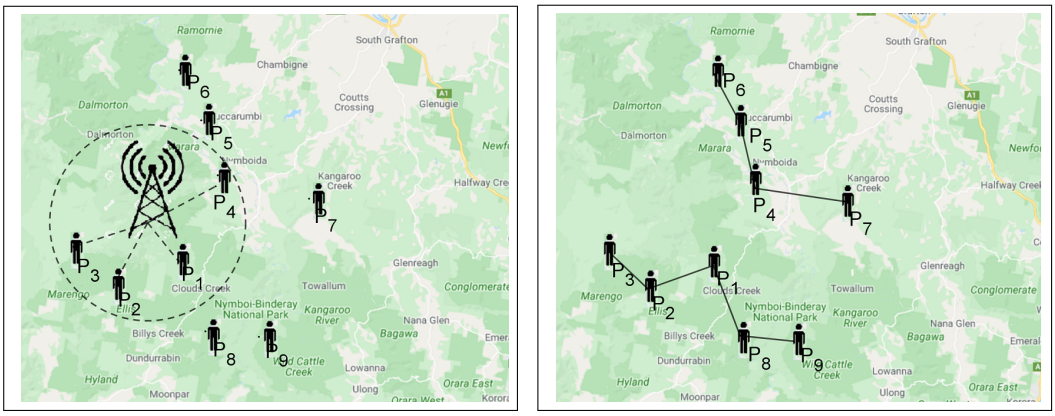
4.1 Overview

In a continuous RNNH query, the points of interest do not change their locations although the queries are continuously moving. We assume a client-server paradigm in peer-peer systems. In most P2P systems, peers are connected by means of a limited range of uniform networks, leading to issues when some connected peers are isolated from the others. In order to address such issues, isolated peers rely on devices with long-range networks to relay their messages. However, since long-range devices can move freely, the set of connected peers may lose their connection. Hence, it is important not only to identify, but also to maximize the area known as the safe region (SR) where a long-range device can move freely while still maintaining connection with its peers. This chapter presents an innovative and generic monitoring framework that addresses the issues related to the frequent updating of query location using a systematic approach. In our approach, we apply the Reverse Nearest Neighbourhood (RNNH) concept in a P2P environment to efficiently identify and maximise the irregularly shaped area of the SR up to four times for the potential movement of the long-range devices. It was found that there is no need for costly re- computation when the query is retained within the SR. Experimental results demonstrate the effectiveness and efficiency of our approach.

4.2 Motivation

The growing importance of location-based services has led to a new range of real-time services such as location-based social networking that uses GPS to locate users and enables them to broadcast their locations and share information via their mobile

devices. Many of these systems have a central base station [6],[7] (see Figure. 4.1(a)); i.e., the entire system relies on one central point. Therefore, if the central point is unavailable or shut down, the whole network becomes unavailable. Another problem is that some points may not be reachable from the base station if they are outside its range, such as p_5 as shown in Figure. 4.1(a). These centralised systems are susceptible to base station failures and isolated points issues.



(a) Centralised systems (b) P2P systems

Figure 4.1: Centralised systems versus P2P systems

In response to the limitations of centralised systems, the emergence of mobile peer-to-peer (P2P) systems offers a promising solution. P2P overcomes the central failure (base station) and isolated points, where points can communicate or exchange information with their reachable surrounding points via short-range wireless communication (e.g. Bluetooth, Ad-Hoc WiFi etc.) [165]. These interconnected peers in a P2P system form a neighbourhood [22]. Figure 4.1(b) shows how a P2P environment can overcome the failure of a centralised base station issue by eliminating the need for such stations and allowing more points, such as p_5 (similarly p_6, p_7, p_8 and p_9), to communicate with the network environment. Although the P2P system has overcome problems related to the centralised base station and isolated points, this method can still cause a new isolation problem such as the isolation of neighbourhoods as illustrated in Figure 4.1(b). Here, the neighborhoods are unable to communicate with each other due to their limited communication range.

Most studies on P2P systems focus on short-range communication systems, leading to the isolated neighbourhoods problem. However, one study has proposed the design of a LoRa wireless mesh network system that combines two types of communication: short-range and long-range [166]. With two types of communication, a system can overcome the problem of isolated neighbourhoods by enabling the two

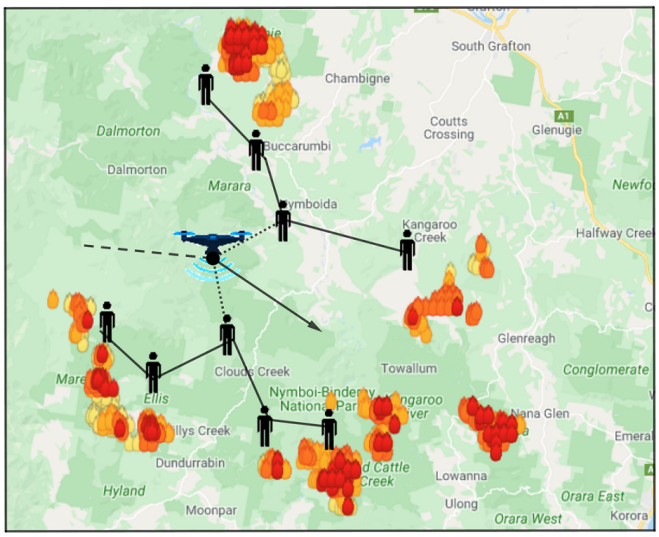


Figure 4.2: Two types of communications in RNNH based P2P network systems

neighbourhoods to communicate with each other through long-range communication. The short-range communication device enables communication among the peers within a neighbourhood. On the other hand, the long-range device serves as a bridge that transfers information between neighbourhoods and ensures that the entire network is connected. This system differs from the base station in a centralized system, in that the base station is solely responsible for transferring information from one point to another point; hence, when a base station is down, the whole network fails.

In the proposed RNNH-based P2P networks, neighbourhood members with short-range communication rely on long-range devices to build a complete network environment as presented in Figure 4.2, where long-range and short-range communications are depicted by the dashed lines and solid lines, respectively. However, the movement of a long-range device away from the short-range members can affect the communication between the short-range members and the long-range device. The long-range device may move to link with other neighbourhoods but, if it moves too far away, it could lose its own neighbourhood members. The main goal, therefore, is to avoid losing the current neighbourhoods when the long-range device moves. To handle this issue, we propose to find a safe region (SR) for a moving long-range device: a boundary within which a long-range device can move freely. Finding an SR for a moving long-range device is an important aspect of keeping its neighbourhoods unchanged and a solution for ensuring that the neighbourhood members remain connected to the long-range device when it moves.

To illustrate the need for SRs in RNNH-based P2P networks, let us consider a real-life example of disaster management involving bushfires which pose great danger to those affected, and need to be handled swiftly and reliably [16]. During the Australian summer, dozens of fires burn across the country and residents brace themselves for catastrophic conditions [17]. In Figure 4.2, people living in bushfire-prone areas often have to leave dangerous areas as a matter of urgency; therefore, support for people in these locations must be maximized. In this kind of natural disaster, local people are disconnected from the centralized base station and communication between people in one neighbourhood can be achieved via Bluetooth or WiFi (short-range device). A resident in that area can take advantage of his short-range device to contact other people within his communication range and find out, for example, the location of the nearest shelter. It is likely that some people in his area already have such information, and that their mobile devices have stored the answer to his query. A moving long-range device operated by the rescuer enables communication between two neighbourhoods and passes instructions and information between victims. The rescuer must therefore move within the limits of a certain region between the neighbourhoods in order to keep in touch with his neighbourhood members. Therefore, in order to manage such disasters, the rescuer must have an SR within which he can move freely in order to make and maintain contact without losing any connection with the neighbourhoods. This scenario demonstrates how two types of communication, namely short-range and long-range communications, in RNNH-based P2P network environments, can assist the affected residents to maintain contact with other residents and with rescuers during a natural disaster.

We summarize our main contributions as follows.

- We introduce the concept of safe region (SR) for continuous query point that keeps the RNNH result unchanged.
- We introduce three types of safe regions: basic, enhanced, and extended.
- We compare the *basic*, *enhanced*, and *extended* safe regions to obtain the maximum safe region within which a query can move freely.
- We conduct an extensive experimental study, applying the Monte-Carlo method to estimate the aggregate area of irregularly-shaped SRs. The performances of the three proposed SRs, the *basic*, *enhanced*, and *extended*, were compared by calculating the total area of SR(s) produced by them.

4.3 Proposed Framework

This study considers a systematic approach that can continuously monitor the moving RNNH query in spatial databases. In this approach, the location of the moving query is known by the server and the query location is updated only if the query leaves its safe region (SR). We propose three types of SRs with different trade-offs within which a query can move without changing its neighbourhood members. An irregularly-shaped area is implemented in this work to represent and to maximize the SR, where the set of objects of interest that are part of the neighbourhood of a query do not change as long as the query point stays within the area. In the case of a bushfire (refer to Figure 4.2), the decision regarding the safe region to have for the rescuer is a trade-off between the size of the safe region and the computation cost. The best safe region will be determined by the most important factors and priorities of the authorities who manage this kind of disaster.

4.3.1 Basic Safe Region

In this approach, the SR of an arbitrary query point (long- range network device) is computed at the server side according to the closest point ($firstclose$) to the query in reverse nearest neighbourhood. After that, the computed SR is sent by the server to the moving query. The objective of the basic SR technique is to construct an SR for the query point which is computationally very fast. In our approach, this basic SR is generated based on two parameters: (i) F_d and (ii) $firstclose$, where F_d is the distance between the nearest competitor facility and the reverse nearest neighbourhood of the query q, while $firstclose$ refers to the closest point of the query q in the reverse nearest neighbourhood. The basic SR refers to the circular region around $firstclose$ with F_d as the radius. The centre of the basic SR is located at $firstclose$ since $firstclose$ is the nearest point of the query in the reverse nearest neighbourhood.

Let us consider the example given in Figure 4.3, which shows the reverse nearest neighbourhood result for the query q, i.e., $NH = \{p_1, p_2, p_3, p_4\}$, where the neighbourhood points are connected by solid lines. In Figure 4.3, F_d is the distance between p_4 and facility E, where facility E is the nearest competitor facility for the reverse nearest neighbourhood NH of q and p_1 is the $firstclose$ of q in the reverse nearest neighbourhood $NH = \{p_1, p_2, p_3, p_4\}$. It is easy to verify that the query point can move freely within the F_d vicinity from p_1 (in any direction) thereby eliminating the need to re-compute of the reverse nearest neighborhood members of q as its reverse nearest neighborhood $NH = \{p_1, p_2, p_3, p_4\}$ stays the same.

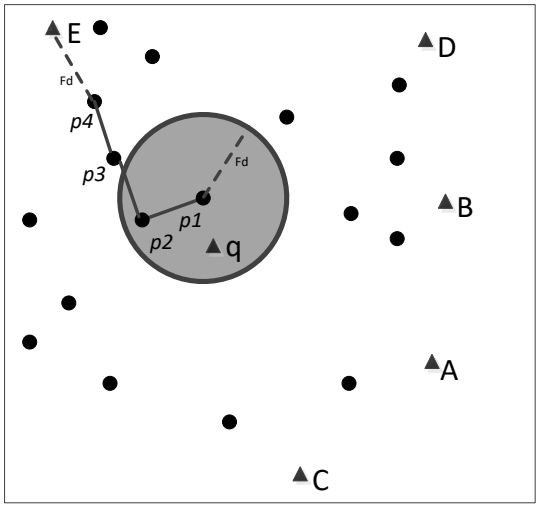


Figure 4.3: Basic safe region

When the query q moves out of the shaded area in Figure 4.3, the current neighborhood NH might no longer be the RNNH for the query point q. This is because the $NH = \{p_1, p_2, p_3, p_4\}$ might become closer to another competitor facility. For example, when the query point moves north by more than the distance F_d from p_1, the NH does not consider q as its closest facility as the distance between p_4 and the facility E is shorter than the distance between p_1 and the query point q (i.e., $d_H(q, NH)$). If the query moves out, the RNNH of the query point is re-computed as $\{p_1, p_2, p_3\}$ as shown in Figure 4.4. If the query point moves out in any direction within the shadowed area as shown in Figure 4.3, for which the radius must be less than F_d, the RNNH i.e., $NH = \{p_1, p_2, p_3, p_4\}$ still considers the query point q as the closest facility and its neighborhood NH does not change.

In summary, the current RNNH of the query remains valid as long as the query stays within the basic SR, i.e., F_d vicinity from the $firstclose$ of q (shadowed area as shown in Figure 4.3). When the query q moves out of the basic SR (exceeding the distance F_d from the $firstclose$ of the query q), the query updates the server with its location information. The server then computes the new SR and RNNH result for the query point q. This new SR is then sent to the moving query q.

Lemma 3 *The shortest (travel) path of a query to change the original set of RNNH points occurs when it moves in a direction opposite to its $firstclose$ to exit the SR.*

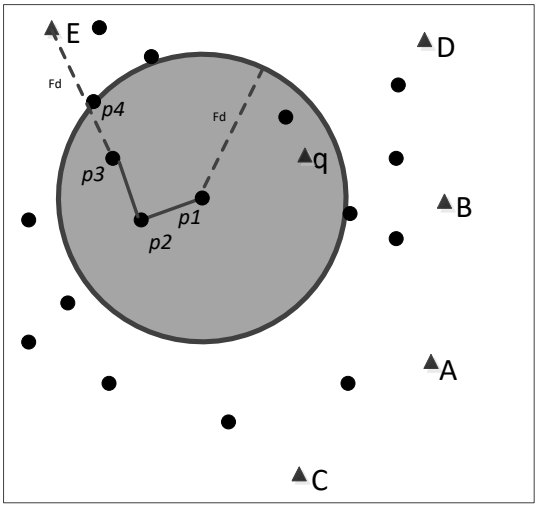


Figure 4.4: RNNH for the current location of q and the corresponding basic safe region

Proof 3 *(By inspection) Since the $firstclose$ object ap- pears to be the point closest to the original query q, moving away from this point in a straight line (in the direction of original query) offers the shortest travel path for the query to exit the SR.*

Lemma 4 *An object is discarded from the moving query RNNH result if its distance from a competitor facility is less than $dist(q, NH)$.*

Proof 4 *Assume that there are four points in the neighbourhood $NH = \{p_1, p_2, p_3, p_4\}$ (as illustrated Figure 4.5). The distance between p_1 and q, $dist(q, p_1)$ (i.e., $d_H(q, NH)$) is lower than the radius (i.e., F_d) of the SR of the query q, F_d. Consider that the query moves to the new location q', which is outside the SR of the current location of q. In this case, the distance between p_4 and the competitor facility E of q becomes less than $dist(q', p_1)$, i.e., $dist(p_4, E) < dist(q, p_1) = d_H(q', NH)$. In this case, p_4 is excluded from the NH of q.*

Lemma 5 *Query point q can move in any direction within the F_d distance from the $firstclose$ of the query q in a reverse nearest neighbourhood NH without being connected to the server to update its RNNH result.*

Proof 5 *Assume that p_1 is the $firstclose$ and the distance between p_1 and q ($dist(p_1, q)$) is less than F_d. Let (q_x, q_y) be the coordinates of the query q and (p_{1x}, p_{1y}) be the coordinates of p_1, which is the $firstclose$, then we get the following.*

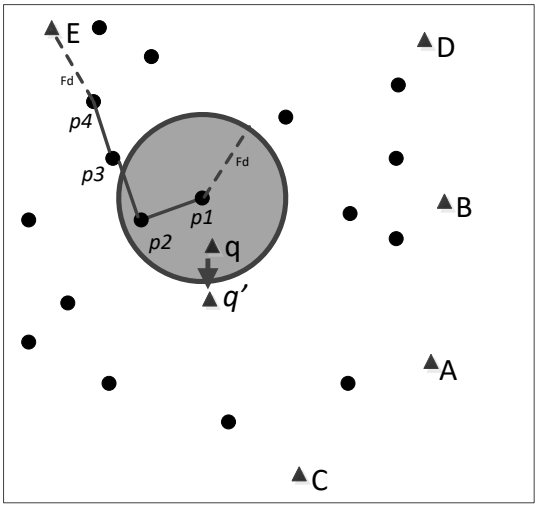


Figure 4.5: Query q moves to q' : $dist(q, p_1) < F_d$ and $dist(q', p_1) > F_d$

$$dist(p_1, q) = \sqrt{(q_x - p_{1x})^2 + (q_y - p_{1y})^2} \tag{4.1}$$

The location of the query point is obtained from $dist(p_1, q)$ *and* F_d *as given as follows.*

$$dist(p_1, q) - F_d = \begin{cases} > 0 & q \quad outside \quad the \quad SR, \\ = 0 & q \quad on \quad the \quad SR \quad boundary, \\ < 0 & inside \quad the \quad SR \quad boundary. \end{cases} \tag{4.2}$$

Based on the definition of RNNH query (on chapter 3), $d_H(q, NH) \leq d_H(f, NH)$, $\forall p \in NH$ *and* $\forall f \in F \setminus q$. *If the query point is on the boundary of SR, two configurations must be taken into account: if q lies inside or outside the SR. If and only if q lies inside the SR,we obtain the following.*

$$(q_x - p_{1x})^2 + (q_y - p_{1y})^2 - F_d^2 \leq 0 \tag{4.3}$$

If q lies outside the SR, then a recalculation of the RNNH is needed. Let us consider the example illustrated in Figure 4.5 for $NH = \{p_1, p_2, p_3, p_4\}$. *If q stays within the SR,* NH *is RNNH for q, i.e.,* $dist(p_1, q)$ *does not exceed* F_d. *However, when* $dist(p_1, q)$ *exceeds* F_d *(e.g., q moves to* q'*), the current* NH *is no longer the RNNH for q.*

One of the consequences of the basic safe-region-based approach is that if the query point moves a distance greater than F_d in the exact direction of the $firstclose$, it will remain within the SR; i.e, if a query is moving towards the $firstclose$ this increases the probability that the query will not leave the SR.

4.3.2 Enhanced Safe Region

The Enhanced SR is an intermediate case between Basic and Extended SRs. The SR can be created more easily through this method by generating a wider SR enabling the query to move more freely. The Basic SR considers a query as being within the SR as its distance away from the $firstclose$ position is less than F_d. Lemma 3 indicates the quickest way that a query can leave the SR is by moving F_d further away in the opposite direction of the $firstclose$ point. However, the query can move further than the F_d distance from the current location and still remain within its RNNH group.

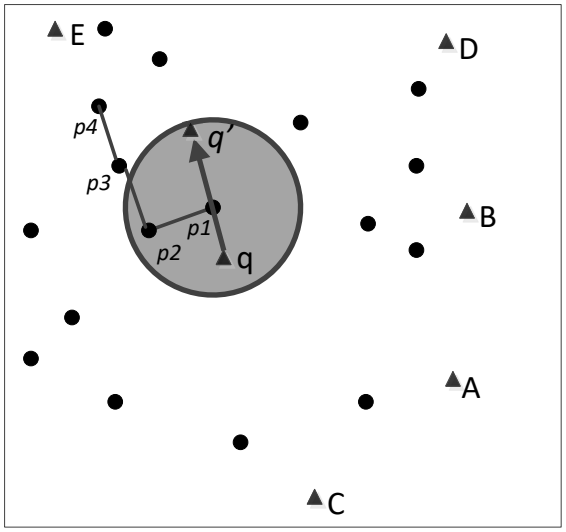


Figure 4.6: Lemma 6

Lemma 6 *Since query point q and $firstclose$ are not in the same position, q can move more than the F_d distance towards the $firstclose$.*

Proof 6 *(By inspection) Figure 4.6, $(q_x; q_y)$ illustrates the location of q. The query (q) can move to a new location (q′) in the required direction of $firstclose$. It still lies within the SR and F_d as the radius. The distance between q and q′ is calculated*

as follows :

$$dist(q', q) = \sqrt{(q_x - q'_x)^2 + (q_y - q'_y)^2} \tag{4.4}$$

Based on Figure 4.6, we can conclude that $dist(q', q) > F_d$.

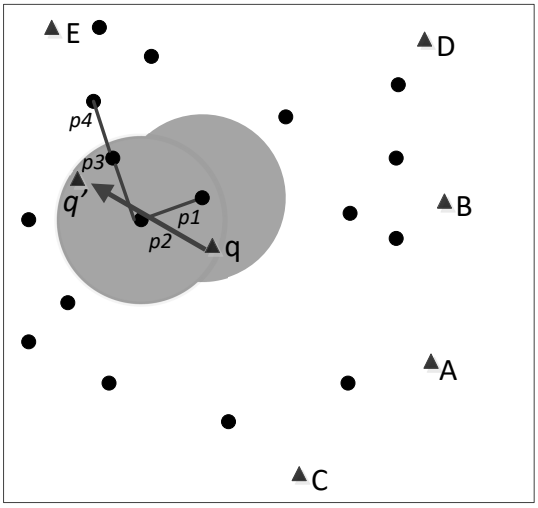


Figure 4.7: Enhanced safe region

The Basic SR may be improved by considering the two points nearest to the query, as well as the distance between the nearest competitor facility and RNNH. Let F_d be the radius of the Basic SR, p_1 and p_2 be the nearest and the second nearest points to q, respectively, and the distance between p_1 and p_2 be less than d. Lemma 3 states that the shortest path that a query takes to change its points of interest is upon moving the F_d distance (in opposite direction) away from the $firstclose$. However, if the query moves a distance of F_d away in the direction of *Secondclose*, it retains its objects of interest.

Lemma 7 *The query point can move more than the* F_d *distance away from the* $firstclose$ *without being connected to the server to update its RNNH result., if it moves towards Secondclose.*

Proof 7 *(By inspection). Based on Lemmas 4 and 6 we can conclude the following observations from the example displayed in Figure 4.7. Assume that* $dist(q', p_1) = P_d$ *and* $dist(q', q) = C_d$ *where* C_d *is greater than* F_d *(*$C_d > F_d$*) and* $P_d > F_d$*. The* $d_H(q', NH) < d_H(f_c, NH)$ *where* $NH = \{p_1, p_2, p_3, p_4\}$ *and* f_c *is the nearest facility*

to NH, which is E. Hence, q can move to q' position C_d, where $C_d > F_d$, towards Secondclose and avoid the connection with the server (see Figure 4.7).

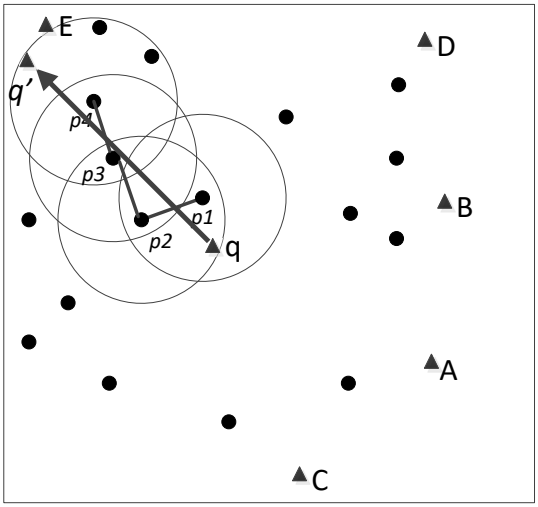


Figure 4.8: Query moves $2F_d$ distance.

In the Basic SR, the centre point is the first point closest to the query, while the Enhanced SR needs to have two centre points, which are the two points closest to the query ($firstclose$, $Secondclose$). Based on observation, the area of the Enhanced SR is greater than that of the Basic SR and q can move more freely than F_d distance from its current location. This is because; the Enhanced SR border is located further away from the query than is the border of the Basic SR, as shown in Figure 4.7.

4.3.3 Extended Safe Region

This section presents the Extended SR, where a query can move further without the need for re-computing the RNNH results. As depicted earlier, the fastest way that a q can leave the SR is by moving away from the $firstclose$ point in the opposite direction, as stated in Lemma 3.

Lemma 8 *A query can move more than $2F_d$ and points of the RNNH (q) remain unchanged if the query moves only towards the points of RNNH with the range of F_d radius.*

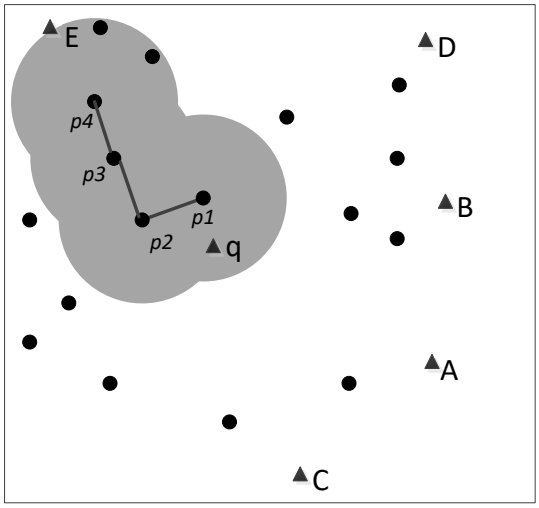


Figure 4.9: Extended safe region

Proof 8 *Assume that RNNH* $= \{p_1, p_2, p_3, p_4\}$*, and the minimum distance between* p_1 *and query point* q *(*$dist(p_1, q)$*) is less than the distance of* F_d*. From Lemmas 4, 6 and 7, we can conclude that query point* q *can move to the position of* q' *(as illustrated Figure 4.8), where* $dist(q', q) > 2F_d$*. In this scenario,* q *does not exclude any points of interest, because the distance from* q' *to RNNH is* $\leq$ *the distances to other facilities (i.e.* $d_H(NH, f_c) > d_H(NH, q')$ *),* f_c *is the nearest facility to* NH*).*

Figure 4.9 depicts an irregular shape that has query point q. More precisely, the Extended SR has a series of round areas, the centres of which are members of the RNNH. Hence, q can move freely in any position inside the Extended SR while holding its RNNH result. This area is dynamically increased and decreased based on RNNH members. This method reduces (1) the frequency of communication between query and server, and (2) the frequency of location updates.

4.3.4 Algorithms for Safe Regions

The following subsection presents further details regarding algorithms for Basic, Enhanced, and Extended SRs.

4.3.4.1 Basic SR Algorithm

Algorithm 3 is used to find the Basic SR where all the neighbourhood-points (RNNH result) listed in the safe-objects list (SOL) which are first found by RNNH algorithm [22]. In line 2, Algorithm 3 calls the calculating boundary function (Algorithm 2).

In algorithm 2, p_i is a point from SOL list. The SOL should be sorted in ascending order from the query point (line 2). SOL may or may not be inside the Z_q, which is the concept of the Voronoi diagram or the Influence Zone introduced by [24]. In line 4, the nearest facility to p_i is determined. In the next step, it checks whether or not it is inside Z_q. If the SOL member is inside Z_q, the processing starts by finding the second nearest facility.

Algorithm 2 Function CalculateBoundary

```
   Input:  SOL (Safe Objects List)
   Output : boundary
boundary ← ∞
sort(SOL,q)
for each p_i in SOL do
    f_c ← NNF(p_i)
    if f_c is Not q then
        d ← dist(p_i, f_c)
        if d < boundary then
            boundary ← d
        end
    end
    else
        f_c ← 2NNF(p_i)
        d ← dist(p_i, f_c)
        if d < boundary then
            boundary ← d
        end
    end
end
return boundary
```

In line 12, when processing SOL members that are inside Z_q, since the nearest facility of SOL members is q, one must find the second nearest facility of p_i. Otherwise, if the nearest facility of SOL members is not q, one must seek the nearest facility to p_i (p_i outside Z_q) and measure its distance to the nearest facility. Lines 6 and 13 calculate the distance between the SOL member and a competitor facility (f_c). If the distance between a competitor facility and SOL members is minimal, then one can place it within the SR boundary (line 8 and 15). This processing continues for all SOL members, then the Calculate Boundary function returns the boundary. Thus, Algorithm 3 obtains the boundary and assigns it to F_d, then the query point (q) is surrounded by a circle with its centre being the closest point to q, which is p_i, with a radius equal to f_d distance (the distance between the RNNH and the nearest

competitor facility). The result of this algorithm is the range circle of the moving query, and its radius is f_d from the nearest neighbour of q.

Algorithm 3 Basic Safe Region Algorithm

Input: q : query point

Output : Basic safe region for q

$SOL \leftarrow RNNH(d, m, q, U, F)$

$F_d \leftarrow CalculateBoundary(SOL)$

$Circle(p_i, F_d)$

$BasicSafeRegion \leftarrow Area(Circle)$

4.3.4.2 Enhanced SR algorithm

Algorithm 4 is used to find the Enhanced SR for a moving query. Here, the SOL is calculated by using the RNNH algorithm [22]. In line 2, after calling Algorithm 4.3.4.1 to calculate the boundary, Algorithm 4 conducts the processing steps as we explained in Algorithm 3, and then should create the Enhanced SR based on the closest two objects of SOL to the query point. The area of Enhanced SR is formed by two circles with a radius of F_d, whose centres are the two closest points to q. In this algorithm, q can freely move further away from the F_d distance from its current location.

Algorithm 4 Enhanced Safe Region Algorithm

Input: q : query point

Output : Enhanced safe region for q

$SOL \leftarrow RNNH(d, m, q, U, F)$

$F_d \leftarrow CalculateBoundary(SOL)$

$Circle_1(p_i, F_d)$ $EnhancedSafeRegion \leftarrow Area(Circle_1)$

$Circle_2(p_{i+1}, F_d)$ $EnhancedSafeRegion \leftarrow Area(Circle_2)$

4.3.4.3 Extended SR algorithm

Algorithm 5 calculates the Extended SR for a moving query. First, the set of points of interest are found (RNNH result) [22] to SOL list. The Extended SR is created with the steps shown in Algorithm 5. In short, it resembles a series of circles. This SR is formed by the overlap of each circular region containing q; their centres are the SOL members. In this algorithm, the query point can move more than $2F_d$ to its current location.

Algorithm 5 Extended Safe Region Algorithm

Input: q : query point

Output : Extended safe region for q

$SOL \leftarrow RNNH(d, m, q, U, F)$

$F_d \leftarrow CalculateBoundary(SOL)$

for *each object* p_i *in* SOL **do**

  $Circle_i(p_i, F_d)$

  insert $Area(Circle_i)$ into $ExtendedSR$ list

end

4.4 Calculating the Area of the Safe Region

When an SR has one or two objects, the calculation of the area is just a straightforward geometric issue [144]. If it is the Basic SR, it is calculated based on the following equation where the radius of the SR is F_d and the centre is the point nearest to the query:

$$Rpi = \pi F_d^2 \tag{4.5}$$

If there are more than two objects, the calculation of SR becomes intricate because the overlapping areas have more irregular shapes. Hence, this study proposes the Monte-Carlo approach. Consider a set $RP = \{Rp_1, Rp_2, ..., Rp_n\}$ of n circles, wherein F_d is the radius and the centres are $\{p_1, p_2, ..., p_n\}$. The circles in RP may partially overlap.

Figure 4.10 illustrates the formation of SRs using four points, $P = \{p_1, p_2, p_3, p_5\}$ within a space E and $RP = \{Rp_1, Rp_2, Rp_3, Rp_5\}$. For instance, $(A2, A3, ..., A10)$ are SRs derived from the overlap/intersection of some points of interest. When the query is within the area of an point that does not intersect with any other area, the point's whole area becomes a SR for query q, such as the Rp_4 area.

$$dist(pi, q) \leq F_d \text{ and } Rpi \bigcap_{x \neq i} Rpx = \theta. \tag{4.6}$$

4.4.1 Calculating the Two Circles

In many circumstances, the area of two points may intersect and the query can fall into either one; Rpi or Rpj. It may also fall within the intersecting area of the two points: $Rpi \cap Rpj$. In this situation,

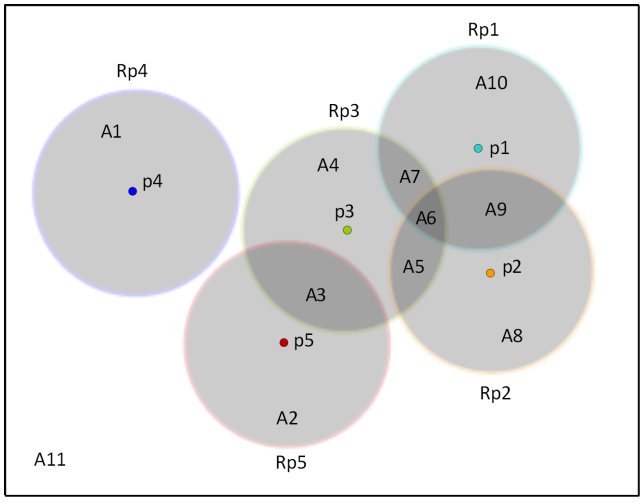


Figure 4.10: Types of extended safe regions

$$dist(pi, q) \leq F_d \text{ and } Rpi \bigcap_{x \neq i} Rpx \neq \theta. \tag{4.7}$$

or

$$dist(pi, q) \leq F_d \text{ and } dist(pj, q) \leq F_d \text{ , } i \neq j. \tag{4.8}$$

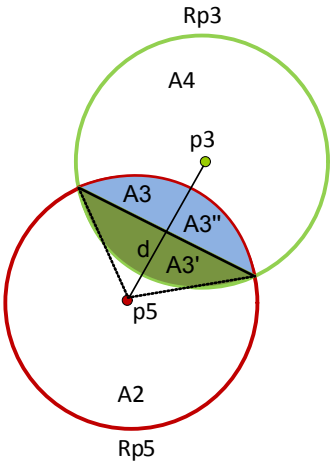


Figure 4.11: Area of intersection of two circles

$A2$, $A3$, and $A4$ (see Figure 4.10) denote the SRs generated by the intersection of two circles. $A3$ has two equal half-regions; $A3' = A3''$. The intersection of the two circles is calculated with Equation 4.7, where d refers to the half distance between two points, and F_d is the area radius, as illustrated in Figure 4.11.

$$Rpi \cap Rpj = F_d^2 arccos(\frac{d}{F_d}) - d\sqrt{F_d^2 - d^2}. \quad (4.9)$$

The calculation of two circles is given by the following equation:

$$\text{Area of Safe Region} = Rpi + Rpj - (Rpi \cap Rpj). \quad (4.10)$$

4.4.2 Using the Monte-Carlo Simulation to Calculate SR Area

When more than two points are involved, the SR may be irregular in shape. Since simple analytical expression is insufficient for calculating the shape areas, the Monte-Carlo simulation [167] can be applied to calculate the area. The calculation of area demands specified queries with the bounding area being a determined size. The multiple points in the bounding area are randomly generated, while counting those within the SR. The area of the SR is calculated as the sum of points in SR, as follows:

$$\frac{\text{sum of points in SR}}{\text{total random points}} \times \text{ area of bounding region} \quad (4.11)$$

4.5 Experimental Results

This section presents experimental results that demonstrate the effectiveness of the proposed algorithms. The algorithms were implemented in C++ and the experiments were run on an Intel Core i7 2.3 GHz PC with 8GB main memory and Ubuntu Linux system. Synthetic datasets were generated to represent the real world [168]. We generate synthetic data based on real data to cover all the patterns and possible scenarios that can occur in real world situations. Three user-point (short-range device) settings with varied densities were created (low=1000 objects, medium=10000 objects, and high=20000 objects) within data space measuring 100 km × 100 km. Table 4.1 summarises the synthetic datasets employed in the experiments. The data points in each dataset are uniformly distributed in the space. Each dataset was indexed by R*-Tree with a node size set to 4096 bytes. Table 5.3 presents the parameters applied to the RNNH query processing algorithm and the default values employed in the experiments.

Three performance measures were adopted in our experiments: (1) the SR area returned using each algorithm; (2) CPU time and (3) memory usage to construct the SR by each algorithm. The performance of the proposed algorithms (basic, enhanced and extended) were assessed by varying the number of facilities (in proportion to the number of user-points) in order to determine the effect of these facilities on the construction of an SR. The densities of the facilities indicate various services offered

Table 4.1: Experiment Dataset

Datasets	**Description**	**# Points**
$SYN1$	Synthetic Low Uniform Distribution	1K
$SYN2$	Synthetic Medium Uniform Distribution	10K
$SYN3$	Synthetic High Uniform Distribution	20K

by the service provider. Several query facilities (long-range devices) were randomly generated for low, medium and high-density environments.

Table 4.2: Experimental Parameters

Parameter	**Range**
Max distance between two points in a NH	2 KM
Min number of member points in a NH	8
Number of points	1K, 10K, 20K
Percentage of facility points	33%, 25%, 20%, 16%

Both the size and the number of SRs (each SR refers to a specific RNNH) derived for the query facility rely on the density of the competitor facilities. These two aspects were varied in subsequent trials. The performances of the proposed safe region algorithms, the average size of the SR and the whole size of the data space, were measured and compared. The purpose of these experiments was to determine and compare the accuracy of the equation-based and simulation-based methods.

4.5.1 Accuracy of Simulation-based Method

An SR was constructed using different object points. The areas of the SRs were calculated using the same methods as those illustrated in Section V, while the Monte-Carlo simulation method is as described in Section 4.4. The area of Basic SR was calculated using F_d values (radii) 5, 6, 7, 8, 9, 10, 11, and 12 Km. The SRs were calculated 100 times using the Monte-Carlo simulation for each query. Then the average of the calculation, was compared with the results obtained by the equation method to determine the accuracy of the simulation method proposed in this study. Figure 4.12 illustrates the area of Basic SR that was calculated using the Monte-Carlo simulation and

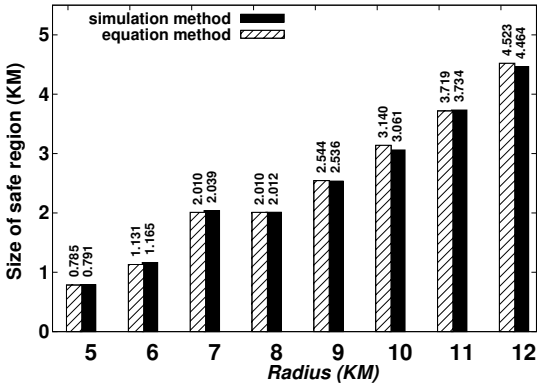
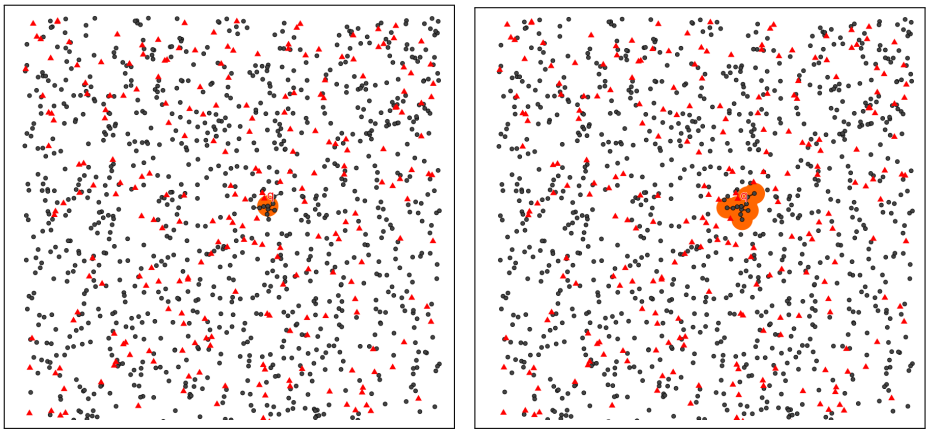


Figure 4.12: Simulation model vs. equation method

equation method. As shown in Figure 4.12 , the results (exact areas) of the simulation method were very close to the results obtained by the equation method (Equation 4.5).

Based on the experiments reported in this section, the bounding region was determined as $100 \times 100 Km^2$, using 100,000 random points to calculate the SR area. To determine the accuracy of the method at these settings, a trial was performed on random queries to calculate the area of Basic SR for a single query with a radius of 2.56 km. In this case, the expected area was 2.56 $\times\pi$ km^2, while the Monte-Carlo simulation estimated the area of the Basic SR accurately to two decimal places at 0.0514 km^2, representing 0.0020 % of the calculated area. Figure 4.13(a) displays a screen shot of the simulation software that was applied to calculate the SR areas to be compared with the whole area.



(a) Basic safe region (b) Extended safe region

Figure 4.13: Demonstration of software calculating the area of safe region corresponding to a static query

Using the same query radius of 2.56 km, the Monte-Carlo simulation estimated the area of the Extended SR to be four times greater than the Basic SR, representing 0.0084 % of the calculated area, as evident in Figure 4.13(b).

4.5.2 Memory Usage

In this section, we show the memory consumed by our Basic, Enhanced and Extended SR algorithms for all the environments with a ratio of 1:4 for the number of user points to facility points. Figure 4.14 shows the memory used by algorithms for the various data set densities: Low, Medium and High. The memory consumption of the safe region is increased with an increase in the size of date set size. Convergence was also noted in the memory usage in Basic and Enhanced SR for all three density environments, because the size of the safe region does not show a big gap between them. Conversely, the memory used by the Extended SR is the largest in all three density environments, because the Extended SR construct needs to store a larger area than do the others.

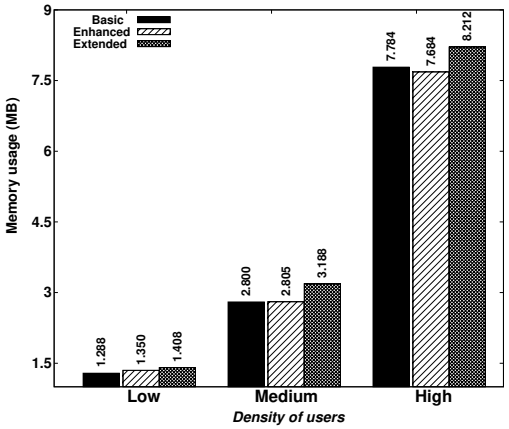


Figure 4.14: Memory usage

4.5.3 Size of the Safe Region

Figures 4.15 - 4.17 illustrate the comparison of the three types of SRs: Basic, Enhanced, and Extended, in three different environments (low: Figure 4.15, medium: Figure 4.16, and high: Figure 4.17) with various numbers of facilities; one to three (33 % of the number of points), one to four (25%), one to five (20%), and one to six (16%). It was discovered that the size of the Enhanced SR was greater than that of the Basic SR (see Figure 4.15). The size of the Extended SR was twice the size of the Enhanced SR. In fact, this was the case for all environment densities, especially in high-density environments (see Figure 4.17). It was observed that the

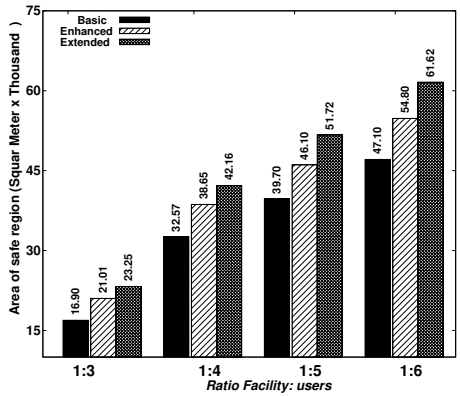


Figure 4.15: Low-density environment

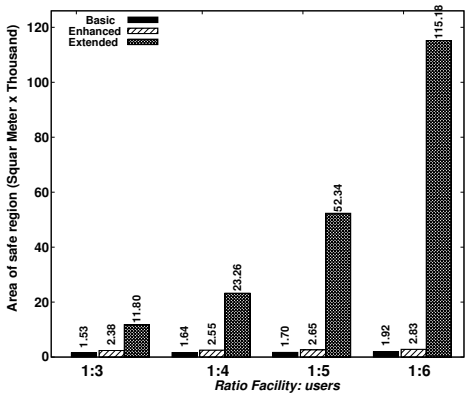


Figure 4.16: Medium-density environment

decrease in the number of facilities affected the SR size. This can be clearly seen in Figure 4.16, where the size of the SR increased subsequent to the reduction in the number of facilities. This is because, when the number of the facility points is lower, the number of RNNH members increases as does the SR area.

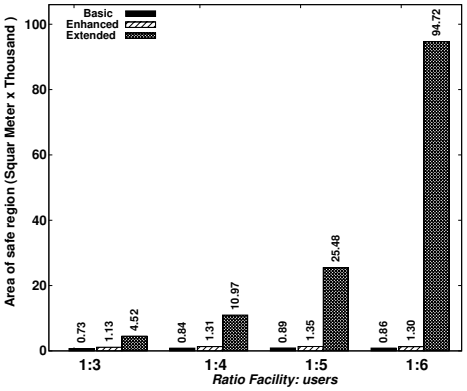


Figure 4.17: High-density environment

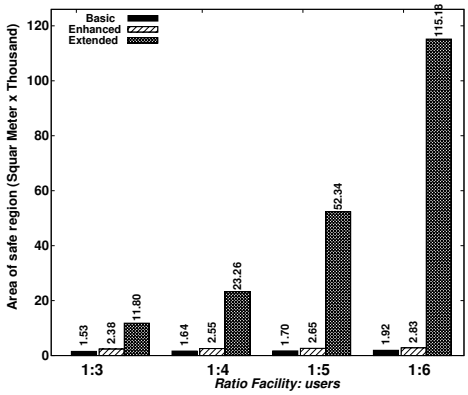


Figure 4.18: Construct CPU time needed for safe regions in three different density environments

4.5.4 Effectiveness of proposed algorithm

A hundred queries were randomly generated in low, medium, and high-density environments, while the SR was calculated using three different types of SRs. The experiment was extended to calculate the execution time for all Basic, Enhanced, and Extended SRs in three density environments. Figure 4.18 displays the execution time for the three types of SRs for all the environments with a ratio of 1:4 for the number of user points to facility points. As observed, the running time was linear with respect to the number of random points used to evaluate the SRs. Convergence was also

noted in the execution time of the Enhanced SR for all three density environments. However, the execution time for the Extended SR was slightly higher than for the Enhanced SR for medium and high densities, while a narrow gap was noted in execution time for a low-density environment.

4.6 Conclusion

In this chapter, we propose a safe-region-based method for continuous RNNH queries. The objective of the safe region is to determine a region within which corresponding query can move freely, while maintaining an unchanged RNNH result. We proposed three safe-region based methods called *Basic*, *Enhanced* and *Extended* using geometric properties to maximize the SRs. The chosen method is a trade-off between the size of the SR and the corresponding computational cost. The experimental results show that the widest area of a safe region is the *Extended* SR, although it requires more computational time than do the other methods. However, while the *Basic* SR requires the lowest computational time, it offers the smallest SR area.

Chapter 5

Reverse Nearest Neighbourhood Queries On Road Networks

5.1 Overview

In a road network environment, the distance measurement depends on the relative positions and road links between two points of interest. The distance between two points is the length of the path from one point to another along the road network. In this chapter, we introduce the concept of reverse nearest neighbourhood for the road network and propose the RNNH-RN algorithm for processing and answering reverse nearest neighbourhood queries on road networks. The performance of the proposed RNNH-RN method has been verified by extensive experiments based on real road network data. Our experiment results demonstrate that our solutions can be feasibly used for networks that have low, medium or high levels of population density.

5.2 Motivation

Spatial databases have emerged as an active area of research in the last two decades. Furthermore, the rapid development of geographic information systems has enabled spatial data query technology to play an increasingly important role in real life. Hence, many applications employ spatial data to help us in some of our daily activities [159]. Examples of popular applications are the Global Positioning System (GPS) [169], targeted marketing and the Location-based Service (LBS) system [8]. Recently, the research focus has been extended to spatial road network data where objects are restricted to a pre-defined path as real road networks contain complex road components such as intersections, curves, and connected and disconnected roads, etc [32].

One of the most common and fundamental queries in the spatial databases of road networks is the reverse nearest neighbour query [86]. The reverse nearest neighbour query is one that given a set of candidate interest points, and a query point, seeks the interest points that consider the query point as their nearest neighbour [85]. For instance, in the example of wholesale distributors and retailers, wholesale distributors normally supply products to retailers, while retailers sell to end customers. In addition to supplying retailers, wholesale distributors deliver their products to the retailers' locations, taking into account the price of the goods as well as the distance between the warehouse and the retailers. It is costly to charge individual delivery fees for each retailer, particularly when they are dispersed. Therefore, we suggest that wholesale distributors might be able to reduce the delivery cost by grouping the orders of all nearby retailers as a single dispatch instead of charging every retailer individually. We believe that our suggestion will reduce the cost and time of delivery.

In a spatial database, all potential retailers that are the nearest to the distributor's warehouse can usually be found by means of an RNN query. As shown in Figure 5.1, the RNN query for query point q returns points A, B, E, I and J as retailers for which q is the nearest facility according to the road network. Although these retailers can be used as one group in the RNN query, they are sparse and treated individually by wholesale distributors. As a result, wholesale distributors are unable to reduce the cost of delivery because they do not consider that the retailers could be geographically close to each other.

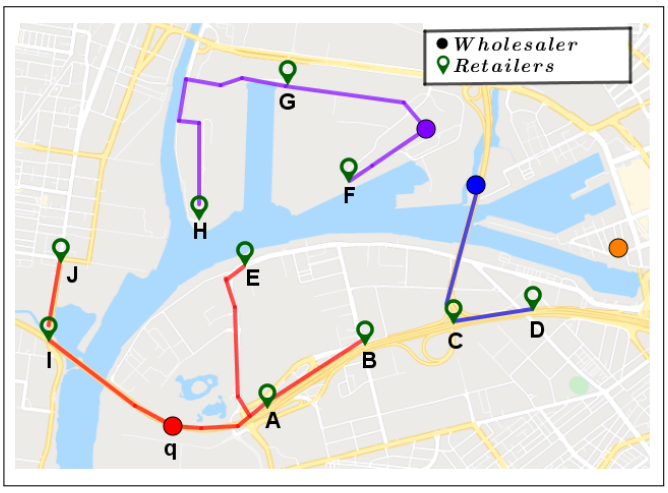


Figure 5.1: Reverse Nearest Neighbour on a road network

To overcome the sparsity problem, we adopt the neighbourhood concept introduced by [23]. A neighbourhood is a collection of m nearby objects where the

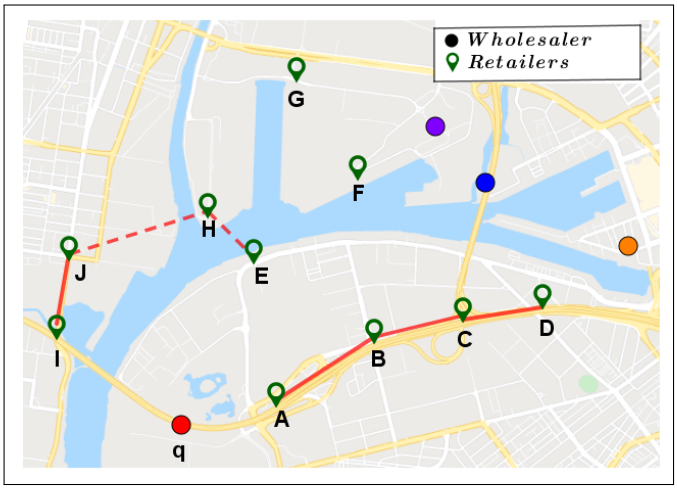


Figure 5.2: An example of euclidean Reverse Nearest neighbourhood

distance between objects must not exceed d. [23] introduced a Euclidean distance-based Reverse Nearest Neighbourhood (RNNH) query. In real life, the Euclidean distance-based approach is not always accurate because it does not consider obstacles such as waterways and buildings. Hence, an authentic road-network-based approach is more suitable in this case as it takes into account actual roadways that include geographical obstacles. Although the approach in [23] is capable of identifying the neighbourhoods of m retail stores, the Euclidean distance is the main drawback that makes this method inaccurate and ineffective in a road network environment as shown in Figure 5.2.

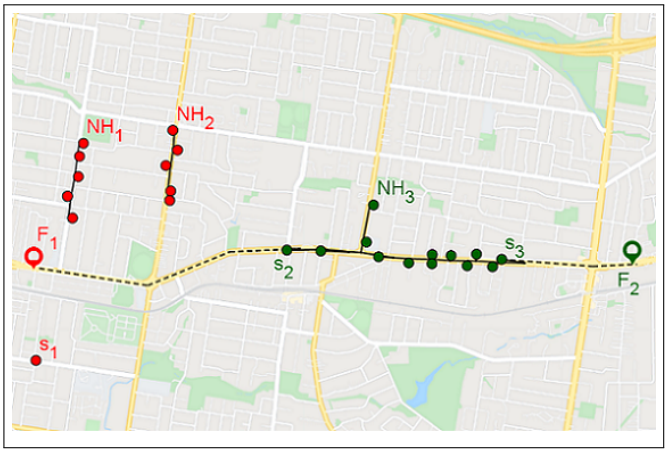


Figure 5.3: RNNH-RN for Wholesale distributor

To manage this issue, we propose Reverse Nearest Neighbourhood on road networks to find the neighbourhood that considers the query point as the nearest query and the neighbourhood members' location is shaped as a chain or strip. Figure 5.3

depicts the Reverse Nearest Neighbourhood for road networks (RNNH-RN). In this example, there are two wholesale distributors: $F1$ is the red distributor and $F2$ is the green distributor. In this example, NH_1 and NH_2 are two separate neighbourhoods of distributor F_1, while retail shop s_1 does not belong to any neighbourhood. Meanwhile, NH_3 is the neighbourhood of distributor F_2. Although, as an individual shop, s_2 is closer to distributor F_1, this shop is considered as part of NH_3 since it is located closer to this neighbourhood. Therefore, shop s_2 will be included in NH_3 where the goods will be delivered from F_2.

In this chapter, we propose an algorithm to answer a Reverse Nearest Neighbourhood on road networks (RNNH-RN) query. Our proposed solutions are based on a real dataset application of road networks and utilize the Network Voronoi Diagram (NVD). To demonstrate the advantages of our proposed algorithm, we conducted a comprehensive experimental study on different environments, varying the density of the real dataset for our proposed algorithm, and compared the results with Euclidean distance outcomes.

5.3 Notations and Definitions

We consider road networks as an underacted weighted graph $G = (V, E)$, where V is a set of vertices and E is a set of edges. A set of points of interest $P = \{p_1, p_2, ..p_n\}$, a set of facilities $F = \{f_1, f_2, ...f_k\}$, and a query point $q \in F$ reside in G. Each edge $(v_i, v_j) \in E$ has a weight (distance travel) [82], which is a positive value. If given a source vertex v_i and destination vertex v_j, the sum of weights of edges along the path is called the distance of the path in the road network, and is denoted by $dist(v_i, v_j)$. Figure 5.4 depicts a road networks. The weight of the edge (P_7, P_9) is 2. The shorter path from q point to F_3 is through $\{P_1, P_4, P_5\}$, which is denoted by $dist(q, f_3) = 5$, i.e. $dist(q, f_3) = (1 + 2 + 1 + 1)$.

The reverse nearest neighbour search, which is one of the fundamental problems in spatial databases has been extensively studied. The reverse nearest neighbour query seeks the points of interest $p \in P$ that consider the query point as the nearest of all facilities $f \in F$. Consider the example given in Figure 5.4, where $\{p_1, p_2, p_3, p_6, p_7, p_{11}, p_{12}, p_{16}\}$ are the only points that consider q as their nearest facility. On the other hand, the nearest neighbour query of an arbitrary point p_i in a dataset F finds the closest point $f \in F$ such that $dist(p_i, f) \leq dist(p_i, f'), \forall f' \in F \setminus f$. The nearest neighbour query of a point p_i is denoted by $NN(p_i)$.

Table 5.1: Notation

Notation	**Definition**
q	The query point
$P = \{p_1, p_2, p_3, ...p_n\}$	A set of points, where n is a positive number
$F = \{f_1, f_2, f_3, ...f_k\}$	A set of facility points, where k is a positive number
p_i	A point p with ID i
NVD_q	The Voronoi cell of the query q
NVD_p	The points located inside the Voronoi cell of the query q
$\|NVD_q\|$	The number of points located inside the Voronoi cell of the query q
d	A network distance parameter between two points
m	A parameter of minimum number of points inside a neighbourhood
NH_i	A neighbourhood consisting of a group of points
$\|NH_i\|$	The number of points located inside the neighbourhood NH_i
f_{p_i}	The nearest facility to p_i
$dist(v_i, v_j)$	The minimum distance between points v_i and v_j
$d_H(p_i, NH)$	The minimum distance between a point (p_i) and a neighbourhood $NH(d, m)$

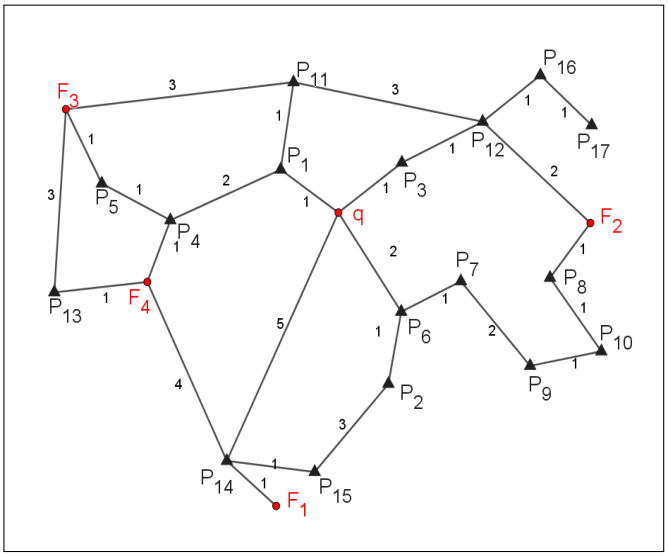


Figure 5.4: A Road Networks

Definition 5.1 ***Network Voronoi Diagram (NVD) of query:*** *Given a set of facilities F, a query facility $q \in F$ and a set of points P, an NVD of q, denoted by NVD_q, retrieving every point p_i in the graph G considers the query facility q as the nearest facility based on the network distance (e.g., shortest-path) between the points. i.e., $\forall p \in NVD_q$, $dist(p,q) \leq dist(p,f)$, $\forall f \in F \setminus q$.*

Consider the dataset given in Figure 5.4. The Network Voronoi Diagram of the query facility q is $\{p_1, p_2, p_3, p_6, p_7, p_{11}, p_{12}, p_{16}\}$. Given its structure and definition, several of the basic properties of the NVD can be obtained. First, the Voronoi diagram has a number of Voronoi cells (VCs). Each Voronoi cell has a generator point (GP) which is located in the centre point of the VCs. Referring to other properties of the Voronoi Diagram, if point p_i is in the Voronoi cell of q, then the distance from p_i to point q is $\leq$ the distance from p_i to other GPs. For more properties, the distance from any point on the edge of the Voronoi cell to the query is equal to the distance of the other generator adjacent to the Voronoi cell.

Definition 5.2 ***Neighbourhood on Road Networks:*** *Given two parameters m and d, a neighbourhood refers to at least m member points where the road networks distance between a member point p_i and the nearest member point p_j in the group does not exceed d. That is, $d(p_i, p_j) \leq d$. The neighbourhood is denoted by $NH(d,m)$.*

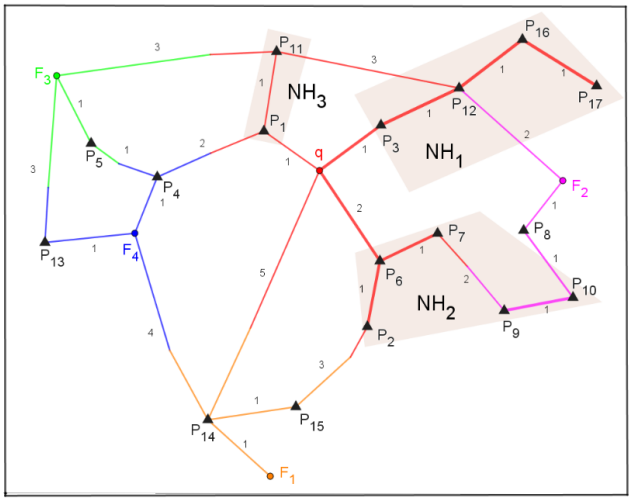


Figure 5.5: Reverse Nearest Neighbourhood query on Road Networks

The neighbourhood $NH(d, m) = \{p_1, p_2.., p_n\} \in P$ refers to a neighbourhood where $1 < m \leq n$ and $p_i \neq p_j$, $\forall i, j \in \{1, 2, ..., m\}$. The distance parameter d denotes the maximum road distance between a member point of $NH(d, m)$. Meanwhile, its nearest member point in $NH(d, m)$ and the cardinality constraint m refer to the minimum number of neighbourhood members. For notational simplicity, here we use NH instead of $NH(d, m)$.

To illustrate the neighbourhood concept, Figure 5.4gives an example of neighbourhood queries in spatial databases. Assume that we are looking for only those neighbourhoods that satisfy the constraints $m = 4$ and $d = 2$. Then, $NH_1 = \{p_3, p_{12}, p_{16}, p_{17}\}$, $NH_2 = \{p_2, p_6, p_7, p_9, p_{10}\}$ and $NH_3 = \{p_1, p_{11}\}$ are neighbourhoods that contain a number of points that have at least four members, with each neighbourhood member having a threshold distance ($d = 2$) to the nearest neighbourhood member. The point p_{11} is not part of NH_1 because the distance between p_{11} and its closest point p_12 in NH_1 is > 2. These points p_1 and p_{11} do not constitute a neighbourhood since there are only two members here, thereby not satisfying the cardinality constraint $m \geq 4$.

Definition 5.3 ***Neighbourhood Distance on Road Networks:*** *Given a neighbourhood $NH(d, m)$ and a point p_i, where p_i is a point of interest or facility, the neighbourhood distance refers to distance on the road network between the point p_i and the closest point in the neighbourhood $NH(d, m)$. It is denoted by $d_H(p_i, NH)$.*

Figure 5.5 illustrates an instance of neighbourhood road network distance, where $NH_1 = \{p_3, p_{12}, p_{16}, p_{17}\}$ is a neighbourhood. Now, $d_H(p_{11}, NH_1)$ is the distance between p_{11} and the nearest member point of NH_1 to p_{12}, i.e. the road networks distance between p_{11} and p_{12} ($dist(p_{11}, p_{12})$). Similarly, $d_H(f_2, NH_1)$ is the road network distance between f_2 and the nearest member point of NH_1 to $p12$; i.e, $dist(f_2, p_{12})$. Again, $d_H(q, NH_1)$ is the road networks distance between q and the nearest point of NH_1 to q, which is $dist(q, p_3)$.

5.4 Proposed Framework

In simplest terms, the RNNH-RN query is a snapshot query that retrieves neighbourhoods $\{NH_i\}$ that consider the query point as the nearest of all the other facilities in the G graph. Consider the static facilities $\{f_1, f_2, f_3, f_4\}$ and the static points $\{p_1, p_2, ..., p_{17}\}$ in the space as depicted in Figure 5.5 . Here, the result of RNNH-RN query for the query facility q is NH_1 and NH_2 for parameters $d = 2$ and $m = 4$ as $d_H(q, NH_1) \leq d_H(f, NH_1)$ and $d_H(q, NH_2) \leq d_H(f, NH_2)$, $\forall f \in \{f_1, f_2, f_3, f_4\}$. The neighbourhood NH_3 is not a reverse nearest neighbourhood of q because $|NH_3| \leq m$. [1]

Definition 5.4 ***Reverse Nearest Neighbourhood Query on Road Networks (RNNH-RN):*** *Given a set of facilities F, a query facility $q \in F$, a set of points P, two constraints d and m, and neighbourhoods $\{NH_1, NH_2, ..., NH_{n'}\}$, a reverse nearest neighbourhood (RNNH-RN) query on road networks for q returns all neighbourhoods $\{NH_i\}$ such that: (i) the road network distance of a point $p_j \in NH_i$ to its nearest neighbour point $p_k \in NH_i$ is less than or equal to d, i.e., $dist(p_j, p_k) \leq d$; (ii) $\forall p_j \in NH_i$, $d_H(p_j, NH_i) \leq dist(p_j, f_{p_j})$, where f_{p_j} is the nearest facility of p_j in F; (iii) $|NH_i| \geq m$; and (iv) NH_i finds the query facility q as their nearest facility among all facilities in F; i.e., $d_H(q, NH_i) \leq d_H(f, NH_i)$, $\forall f \in F \setminus q$, it is denoted by $RNNH\text{-}RN(q, d, m, P, F)$.*

In this section, we explain our solution for processing a Reverse Nearest Neighbourhood on Road Networks (RNNH-RN) query. Our algorithm consists of two phases namely filtering and verification. The goal of the filtering process is to prune and optimize the processing the data point set, and then remove a large number of meaningless points [170, 171]. In this phase, we prune the search space instead of accessing all the nodes. Given a facility f and point p_i, f is a competitor facility of q. A point p_i cannot be the answer for RNNH-RN of q if $d_H(p_i, NH_i) > dist(p_i, f)$. In

[1] It should be noted that there could be zero or more reverse nearest neighbourhoods of a query facility q in a given dataset $P \cup F$ as opposed to the nearest neighbourhood

this case, we say that facility f prunes the point p_i. So, in the range query method, given a set of facilities F and a distance constraint d, a range query is denoted by $rangeQ(q, d, F)$, where $p_i \in P$, returns all the points that are less than or equal to the road network's distance from q. The search stops when it encounters any competitor facility in the G space; even if it does not reach the d distance, the search stops even if the road network's distance to f is less than the d value. Additionally, in the case where NVD_q is empty ($NVD_p = \emptyset$), then there is no reverse nearest neighbourhood for q. Therefore, the processing must stop and not proceed to any further nodes to answer the RNNH-RN query. Hence, in this case, the processing of RNNH-RN stops at a very early stage when $NVD_p = \emptyset$

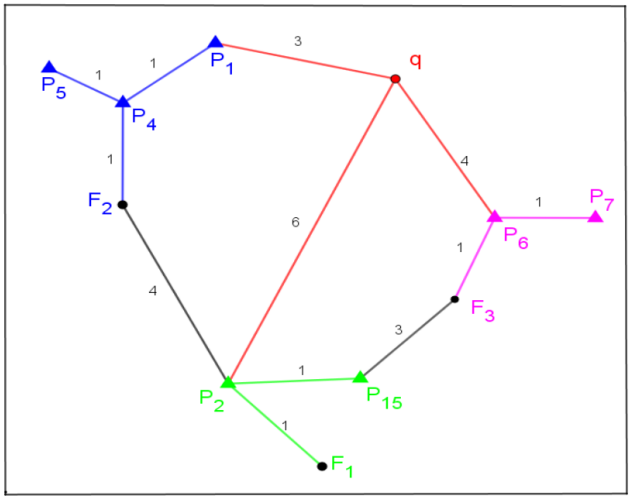


Figure 5.6: Lemma 1

Lemma 9 *If the NVD_q of a query is empty, then there is no reverse nearest neighbourhood on road networks for q.*

Proof 9 *Given facilities F and users P such as $q \in F$ and $\forall p_i \in P$. The NVD_q is created between the q and other facilities. There is no RNNH-RN result, if there are no points within NVD_q, which means that $NVD_p = \emptyset$. In this case, none of the points can consider the q as the nearest facility. Hence, the nearest point p_i to q does not consider the q as the nearest facility. (i.e., $dist(p_i, f) < dist(p_i, q)$). Consequently, in this case, we will not have $d_H(q, NH_i) \leq d_H(f, NH_i)$, as all the nearest points to q do not consider q as the nearest facility. This is because $\forall p_i \in P$, $dist(p_i, q) \not\leq dist(p_i, f)$. As shown in Figure 5.6, there is no point located in the NVD for q, $NVD_q = \emptyset$, the nearest points to q is P_1, and $dist(p_1, f_2) < dist(p_1, q)$.*

Also, there is no need to check the validation process for any point inside the Voronoi Network diagram (NVD) of q. This is because this point must answer the RNNH-RN query with no conditions. The following lemma shows how the point is processed if it is inside NVD_q; it does not require extra validation or verification processing.

Lemma 10 *For any retrieved point p_i where $p_i \in NVD_q$, an entry p_i is a candidate object for reverse nearest neighbourhood query on road networks if the distance between p_i and neighbourhood (NH)is $\leq d$.*

Proof 10 *To prove that p_i is a candidate entry, based on one of the properties of the Voronoi diagram as mentioned earlier, if the q is the generator point of the Voronoi cell, then the road network's distance from q to p_i is less than the distance to any other generator point (facility). Consequently, any points in the Voronoi cell of q considers q to be the nearest facility among any other facilities, which means that $dist(p_i, q) < dist(p_i, f_i)$. Hence, we do not need to check the validation of points in NVD_q, because any point, such as point P_1 and P_2 that are there have already been validated as per definition 5.4.*

For the verification phase, a neighbourhood NH must satisfy this: $d_H(q, NH) \leq d_H(f, NH)$, $\forall f \in F \setminus q$ to be a RNNH-RN of q. Figure 5.5 shows an example of the RNNH-RN query result. A naïve RNNH-RN query processing algorithm first computes a neighbourhood. Then, the algorithm calculates the distance between the neighbourhood and the query, which is $d_H(q, NH)$. Next, for each neighbourhood member $p_i \in NH$ the algorithm calculates $dist(p_i, f_{pi})$, where f_{pi} is the nearest facility to p_i. If $d_H(q, NH) \leq dist(p_i, f_{p_i})$ for all $p_i \in NH$, then the NH is the RNNH-RN of the query q. The problem with this naïve algorithm is its run-time complexity. First of all, the algorithm has to enumerate all possible neighbourhoods NH that satisfy the distance and cardinality constraints d and m, respectively. Therefore, the naïve algorithm is not a viable and suitable solution for RNNH-RN query processing.

In order to overcome the issue associated with the naïve query processing algorithm, we propose an algorithm that can perform better for reverse nearest neighbourhood on road networks. Figure 5.7 shows the main steps for the Reverse Nearest Neighbourhood on the road networks framework. First, the data is prepared by placing the points located in the Voronoi cell of the query in a heap. Then, checking each entry if satisfy the condition of neighbourhood. This process repeated

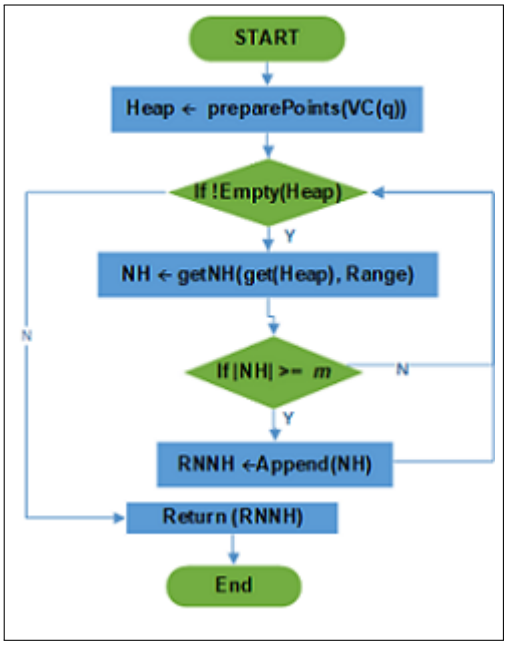


Figure 5.7: Reveres Nearest Neighbourhood framework

until all candidate points have been processed. The algorithm details can be seen in Algorithm 6.

Algorithm 6 is used to find the reverse nearest neighbourhood on a road network which is first computed on a network Voronoi diagram [12, 117] of the query facility q. All points located inside the NVD of q finds the q nearer than any other facility $f \in F$. The algorithm then retrieves all points $p \in P$ that are located inside the NVD of the query; the points list is denoted by NVD_p. Consequently, any points in NVD_q must be part of the RNNH-RN answer, and do not require any extra validation process.

Upon creating the NVD_q for the query facility q and to retrieve the corresponding NVD_p, the algorithm places the points into a min heap to q (line 5). Then, the algorithm begins to generate the first neighbourhood by retrieving the point nearest to q (line7). It should be noted that there could be more than one candidate in NVD_p with the same distance to q, so we randomly pick one of them to break the tie. Line 11 calculates $d_H(q, NH)$ which is the road network distance between q and the nearest point. Next, the algorithm inserts point p_i into a temporary list and retrieves all points from P close to the p_i with a road network distance less than or equal to d. Line 15 checks whether the point is inside the NVD of q; then it is added to the result set (line 17) based on Lemma 10.

Algorithm 6 Computing Reverse Nearest neighbourhood on Road Networks

Input: P: set of points, F: set of facilities, d: distance constraint, m: minimum users and q: $q \in F$

Output: $RNNH\text{-}RN(d, m, q)$

$NVD_q \leftarrow$ Calculate Voronoi Cell of q

$NVD_p \leftarrow$ insert all points in NVD_q

$h \leftarrow$ sort(NVD_p,q)

while h *is not empty* **do**
  de-heap an entry p_i
  if $isNotVisited(p_i)$ **then**
    mark p_i as visited
    initialize $NH_i \leftarrow \{p_i\}$
    $d_H(p_i, NH_i) \leftarrow dist(p_i, q)$
    $vLst \leftarrow rangeQ(p_i, d, P)$
    while $vLst$ *is not empty* **do**
      de-heap an entry c_i
      if $c_i \in NVD_p$ **then**
        mark c_i as visited
        $NH_i \leftarrow append(c_i)$
        $vLst \leftarrow vLst \cup rangeQ(c_i, d, p)$
      end
      else if $dist(c_i, NH_i) \leq dist(c_i, f_{c_i})$ **then**
        if $d_H(p_i, NH_i) \leq dist(c_i, f_{c_i})$ **then**
          $NH_i \leftarrow append(c_i)$
          $vLst \leftarrow vLst \cup rangeQ(c_i, d, P)$
        end
      end
      if $|NH_i| \geq m$ **then**
        $RNNH\text{-}RN(d, m, q) \leftarrow append(NH_i)$;
      end
    end
end

For every point, if the nearest facility is not q and its road network distance to its nearest facility is greater than $d_H(q, NH)$, and its road network distance to the current NH is less than its distance to its nearest facility, it is added to the list, otherwise, it is discarded (line from 20 to 23). The points in the list are marked as visited (line 9 and 16) to prevent any redundant processing. This process is repeated for all unvisited points in the list and the algorithm stops constructing the current

neighbourhood when no more points can be added to the list. If the list has at least m points, then it becomes the first RNNH-RN of q; otherwise, it is discarded (Line 26). The algorithm then retrieves the next point from NVD_p that has not been included in any previous neighbourhood and continues to find the corresponding neighbourhood for it. The above neighbourhood construction process is continued until NVD_p becomes empty.

5.5 Experimental Results

This section presents the experimental results that demonstrate the feasibility of applying the proposed algorithm. The algorithms were implemented in C++ and the experiments were run on an Intel Core i7 2.3 GHz PC with 8GB main memory and the Ubuntu Linux system. We examine Reverse Nearest Neighbourhood on road networks queries on real-world road network graphs.

Table 5.2: Experiment Dataset

Description	**Density level**	**Users**	**Facilities**
Map A : University campus area	Low	60	12%
Map B: A medium-sized city	Medium	690	3%
Map C: Densely populated of suburbs	High	1864	3%

We evaluate our proposed algorithm by simulating scenarios ranging from low-density to high-density in terms of the number of objects in road networks. The network covers all types of roads, including local roads and contains real edge weights for travel distances. We conduct in-depth studies based on datasets for a university campus (low density), a medium-sized city and a densely popu- lated group of suburbs (high density). These datasets are listed in Table 5.2.

In our experiments, we use the term 'low-density' if the number of objects is less than one hundred; we use 'high-density' if the number of objects is greater than one thousand [34]. 'medium-density' is used when the number of objects is deemed to be between one hundred and one thousand. The purpose of our experiments is to show how different factors such as the number of objects determining density and the value of d and m could affect the results of our algorithm in terms of the number of queries obtained, the number of neighbourhood members, and the size of the neighbourhood.

Table 5.3: Experiment Parameters

NH Parameter	**Range**
d (km)	0.1, 0.3, 0.5, 1, 1.5
m	2, 4, 6, 10, 16

In this chapter, several evaluations of the flexibility of the proposed algorithm and the correctness and accuracy of results are evaluated in order to examine the results of the experiments. We evaluate the isolated points of interest and the flexibility of our proposed algorithm to show their tolerance to the constraint of neighbourhoods in comparison with the Voronoi cell based on the Reverse Nearest Neighbour (RNN) algorithm, and the proposed algorithm is proven to be not mutually exclusive. Also, to evaluate the accuracy of the results and the effectiveness of our approach, we compared the Euclidean and road network outcomes to evaluate the effectiveness of our approach to road network queries. We proved that our results are more reliable because they are based on real datasets. Table 5.3 presents the parameters applied to RNNH-RN queries in the experiments.

Table 5.4: Experimental Analysis for Nearest Neighbour - Monash University-

Distance from	**Number of shops**
1-100 m	34
200-300 m	2
300-400 m	7
400-600 m	7
600-700 m	2
700 m - 800 m	2
800 m - 900 m	2
above 1 km	4
The maximum distance : 2.11 km **Average distance to nearest shop: 0.3011 km**	

5.5.1 Low-density Experiment

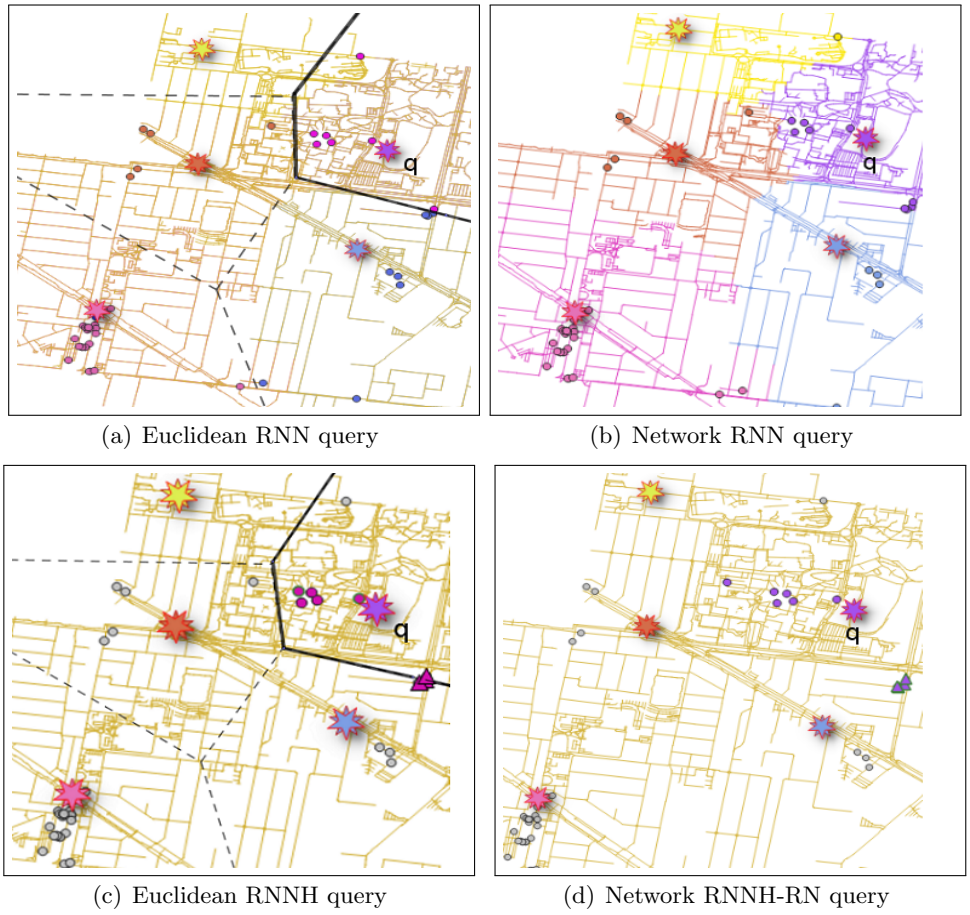
In this experiment, as shown in Figure 5.8, we use the Monash University Clayton campus and residential area as the dataset. We focus on the campus environment which consists of a low-density dataset to test the algorithm with a variety of values for maximum road networks distance (d) and the minimum number of neighbourhood members (m) value (e.g low d and high m value).

Figure 5.8: Monash University -Clayton Campus-

This dataset has approximately 6600 edges and 8100 vertices, and comprises a complex of fifty-two shops and stores. The dataset has an area of 5.20 km^2. We evaluate the range of maximum road network distance (d value) and a minimum number in the neighbourhood in order to determine their impact on the number of queries obtained, the number of neighbourhoods and the size of neighbourhoods members concerning each result of the RNNH-RN algorithm.

In this study, we conduct extensive geospatial analysis to achieve a comprehensive understanding of the behaviour of a low-density dataset. We calculate the network distance from each point of interest (POI) to the nearest neighbour as shown in Table B.6. In the graph, the average network distance to the nearest neighbour is 0.3 km and the maximum network distance to the nearest neighbour is 2.11 km. The total road network distance to the nearest neighbour in the graph is 16 km. As shown in Table B.6, most points of interest are less than 100 m to the nearest neighbour, and only four points of interest have more than 1 km road network distance to the nearest neighbour.

Figure 5.9 shows the difference between and comparison of the outcome of Euclidean distance and of road networks, for the RNN query and reverse nearest



(a) Euclidean RNN query (b) Network RNN query

(c) Euclidean RNNH query (d) Network RNNH-RN query

Figure 5.9: Spatial query results on Map A

neighbourhood (RNNH) query for the purple query, and the RNN query and the RNNH query for the purple query. The result of the RNN query for Euclidean distance is different from that obtained for road network distance: it has almost the same number of neighbourhoods, but it has different neighbourhood members as shown in Figure 5.9(a) and Figure 5.9(b). Based on the spatial network, the travelling distance would be much further.

Figure 5.9(c) displays an example of Euclidean reverse nearest neighbourhood that returns two neighbourhoods (five and three members), which differs from the reverse nearest neighbourhood on the road networks query which returns the results for two neighbourhoods (six and three members). The first neighbourhood has six members on road networks. The Euclidean RNNH does not involve the sixth point because the Euclidean distance from the point to the competitor facility is a shorter distance to the query point. Figure 5.9(d) shows the results of the RNNH-RN purple query pertaining to two neighbourhoods (depicted by circle and triangle).

In terms of minimum value in neighbourhood (m), when the value changes and increases from 4 to 6, the result excludes one neighbourhood based on the query preferences as shown in Figure 5.10. Also, our findings show how the result changes according to the d value as depicted in Figure 5.11. Figure 5.11(a) shows an example of a reverse nearest neighbourhood query when the maximum road distance between neighbourhoods is 0.300 km: it returns one neighbourhood and excludes the points that are more than 0.300 km distant from neighbourhood members.

(a) $m = 4$ (b) $m = 6$

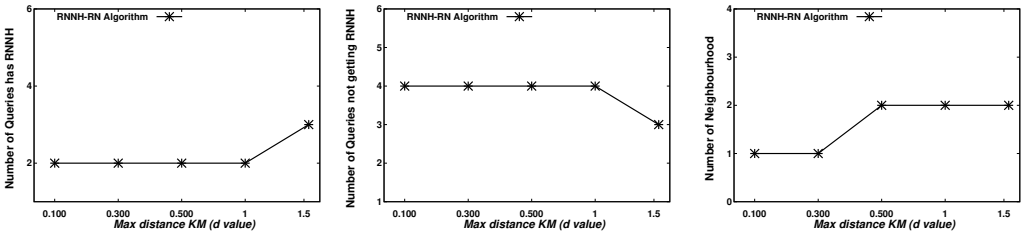
Figure 5.10: RNNH-RN result for $d = 0.5km$

(a) $d = 0.300$ (b) $d = 0.500$

Figure 5.11: RNNH-RN result for $m = 6$

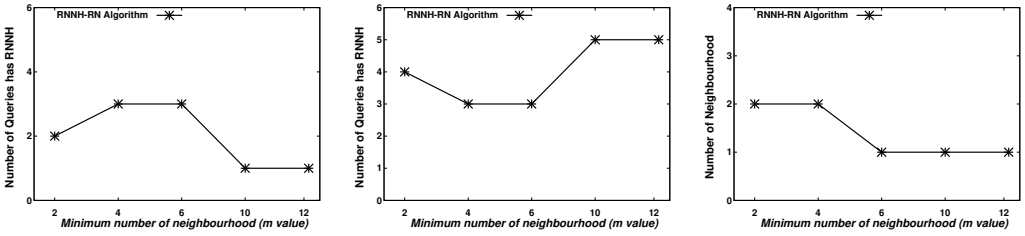
We studied the behaviour of the reverse nearest neighbourhood on road networks (RNNH-RN) with low density, varying the value of the maximum network distance between neighbourhood members (d), and the minimum number of neighbourhood members (m). First, we studied the effect of the d factor on the algorithm. Specif-

ically, Figure 5.12 shows the number of queries obtaining neighbourhood and the average of neighbourhood members in each neighbourhood for varying values of (m). Figures 5.13(a) shows the number of queries that obtain results for the RNNH-RN algorithm. When the maximum road distance (d value) between neighbourhood members is less than 1 km, only two queries obtain the RNNH-RN result, increasing to three queries when $d = 1.5$ km. The reason for this is that RNNH-RN is obtained depending on the closeness of points of interest forming the neighbourhood. Therefore, the chance of obtaining a query result improves when d increases. Our report shows displays the average number of neighbourhoods in each query; we find that the average number of neighbourhoods is one neighbourhood when the maximum network distance is between 0.100 and 0.300 km, and increases to two neighbourhoods when the maximum networks distance increases.



(a) Number of queries with RNNH (b) Number of queries without RNNH (c) The average number of neighbourhoods

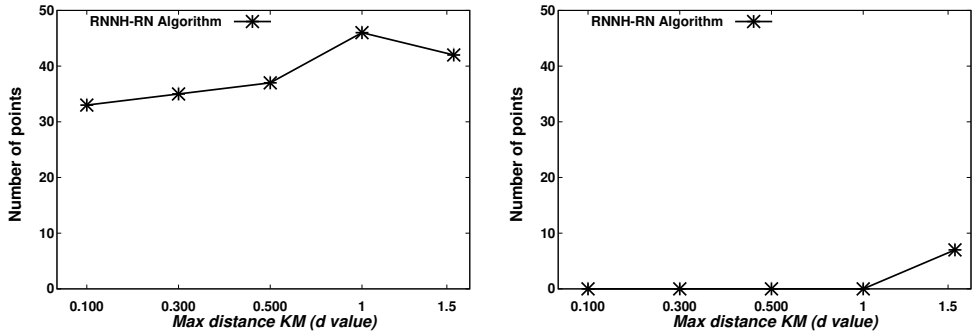
Figure 5.12: Variety of d values



(a) Number of queries with RNNH (b) Number of Queries does without RNNH (c) The average number of neighbourhoods

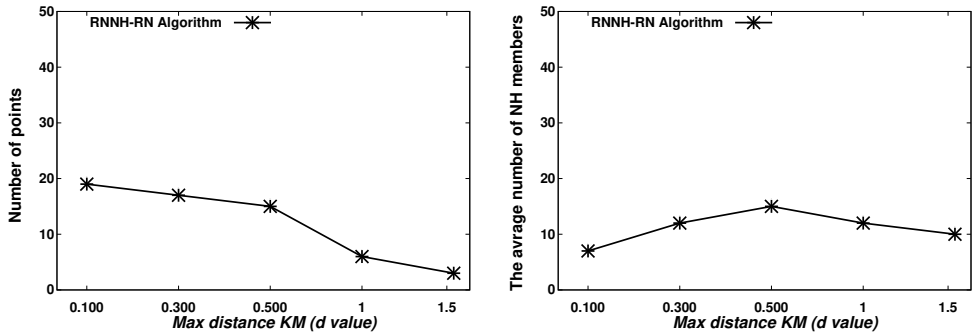
Figure 5.13: Variety of m values

We also study the effect of the minimum neighbourhood number of members (m) in the low-density dataset. We find that, generally, when the value of m increases, the number of queries for neighbourhood decreases dramatically, and the average number of neighbourhoods decreases. The reason for this is that when the minimum number of neighbourhood members increases, neighborhoods with a small number of



(a) The number of points belonging to only one neighbourhood

(b) The number of points belonging to more than two neighbourhoods



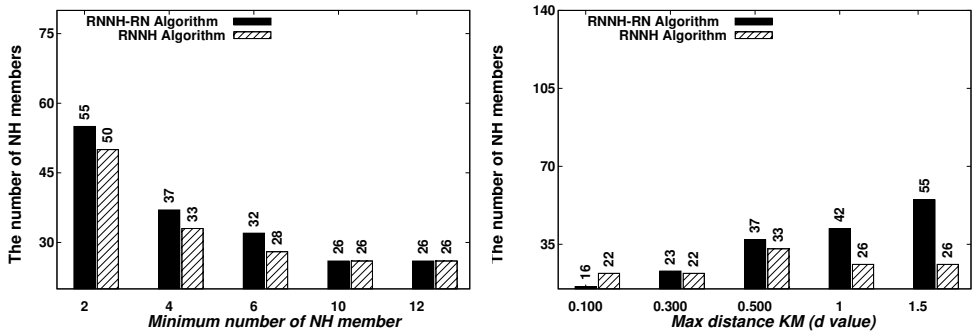
(c) The number of points not belonging to any neighbourhood

(d) The average number of neighbourhood members

Figure 5.14: Analysis for Monash University campus

members are excluded. Figures 5.14 illustrates the average number of neighbourhood members, the number of points belonging to one neighbourhood and the number of points belonging to more neighbourhoods. The number of neighborhoods increases dramatically when the d value is increased. This is unlike the number of points not belonging to any neighbourhood where the number decreases incrementally with an increase in the d value.

Figure 5.15 shows a comparison between reverse nearest neighbourhood in Euclidean distance and in road networks to show the accuracy obtained for road networks distance. In Figure 5.15(b), we see the effect of different values of m and d. It is clear that for both, the number of neighbourhood members decreases dramatically with an increase in m. This is because there is a correlation between the number of neighbourhood members and the m value: when m increases, neighbourhoods that do not satisfy the m value are excluded. We also find that the efficiency of the RNNH-RN query processing algorithm in road networks improves when the d increases and the neighbourhood members in Euclidean RNNH are fewer than



(a) m value (b) d value

Figure 5.15: Comparison RNNH and RNNH-RN algorithm for Monash University campus

RNNH-RN in road networks. This is because for each query, the Euclidean distance from a point to competitor facilities is mostly shorter compared to the distance in road networks, which also increases the number of neighbourhood members.

5.5.2 Medium-density Experiment

In this experiment, we use a spatial database of Melbourne city as the dataset. Melbourne is located on the south coast of the state of Victoria, Australia, and it is divided into two parts, with a medium density of roads in the centre, and light density on the western and eastern sides. For this experiment, we use the main roads that connect the western and eastern points. We use the supermarkets dataset for Melbourne, represented by the grey dots in Figure 5.16.

This dataset covers 2150 km^2, and contains more than 315500 vertices and 321107 edges. It contains approximately 700 supermarkets, which represent points of interest, while eighteen facilities represent suppliers. We conduct a geospatial analysis to determine the behaviour of the reverse nearest neighbourhood algorithm when applied to road networks in Melbourne. network.

By means of a graph, we conduct a geospatial analysis of the nearest points, which are shown in Table B.12. There are 350 points with a network distance of less than 0.100 km to the nearest point, and there are thirty-five points of interest (POIs) between 0.100 km and 0.200 km, and thirty-eight points of interest between 0.200 km and 0.300 km.

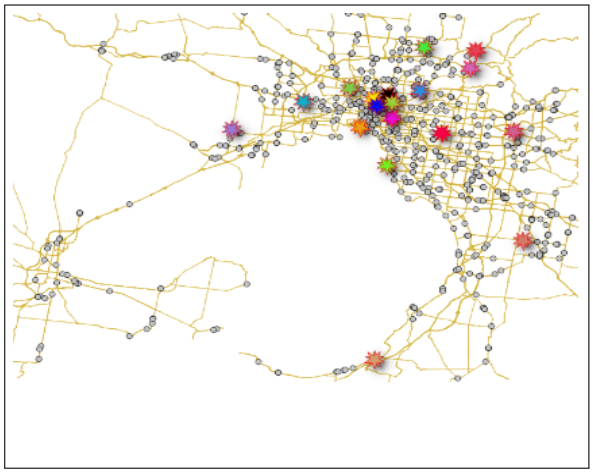


Figure 5.16: Melbourne City, Australia

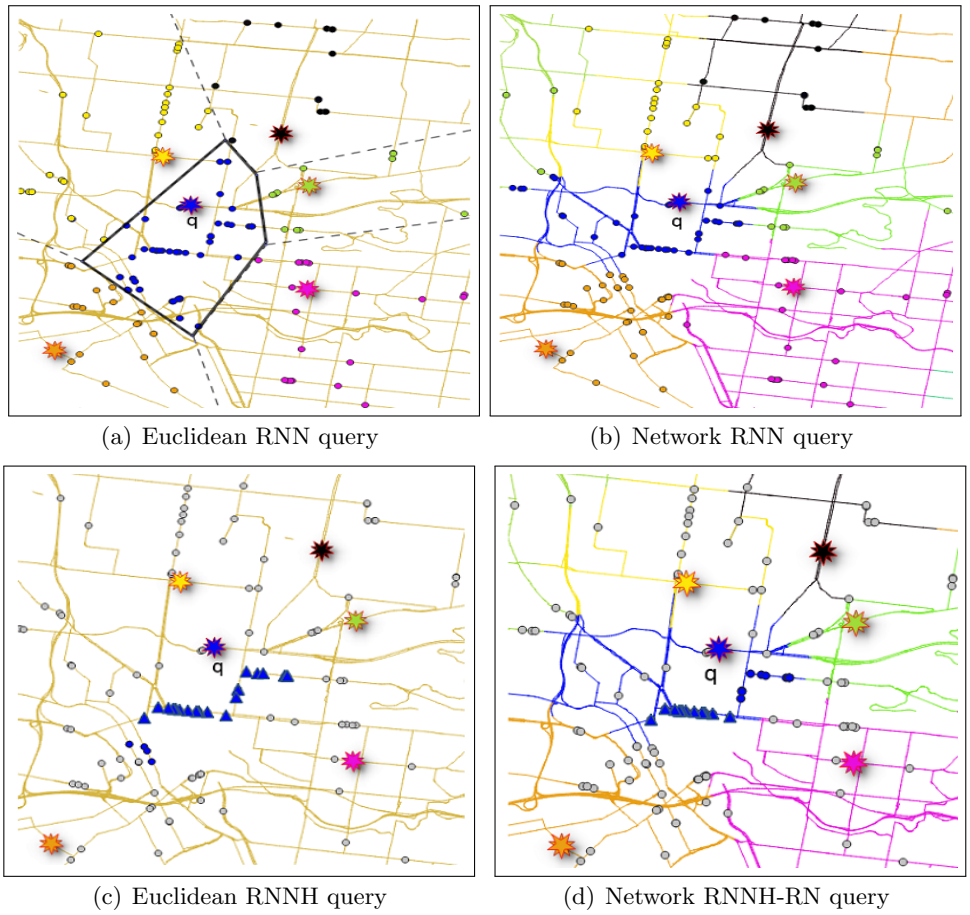
Also, twenty-five points are between 0.300 km and 0.500 km, and forty points are between 0.500 km and 0.800 km. There are two hundred and sixty points that are more than one km from the nearest neighbour. The average distance to the nearest point is 750 m. The graph shows that the total network distance travelled to the nearest neighbour is 2092 km. Tables B.12 gives more details about the experiment conducted for Melbourne city.

Table 5.5: Experimental analysis Nearest Neighbour -Melbourne City-

Distance from	**Number of shops**
1- 100 M	350
100-200 M	29
200-300 M	38
300-400 M	10
400-500 M	13
500-600 M	10
500-800 M	20
800m - 1 km	14
above 1 km	206
The maximum distance : 70 km **Average to nearest shop: 0.750 km**	

In this study, we randomly select the location of queries on different sides. They are marked by stars as shown in Figure 5.17, and we take the four queries (blue, pink,

yellow, green stars) as examples to explain our findings and observations. Figure 5.17(a) shows the Euclidean Voronoi diagram (VD) for four facilities and the outcome of the RNN query for the blue query, while Figure 5.17(b) presents the network Voronoi diagram (NVD). The RNN in the road network excludes several points, and the road network takes into consideration the road distance which is somewhat greater than the Euclidean distance. The blue neighbourhood member belongs to the blue query, the yellow neighbourhood belongs to the yellow query, and the pink neighbourhood belongs to the pink query.



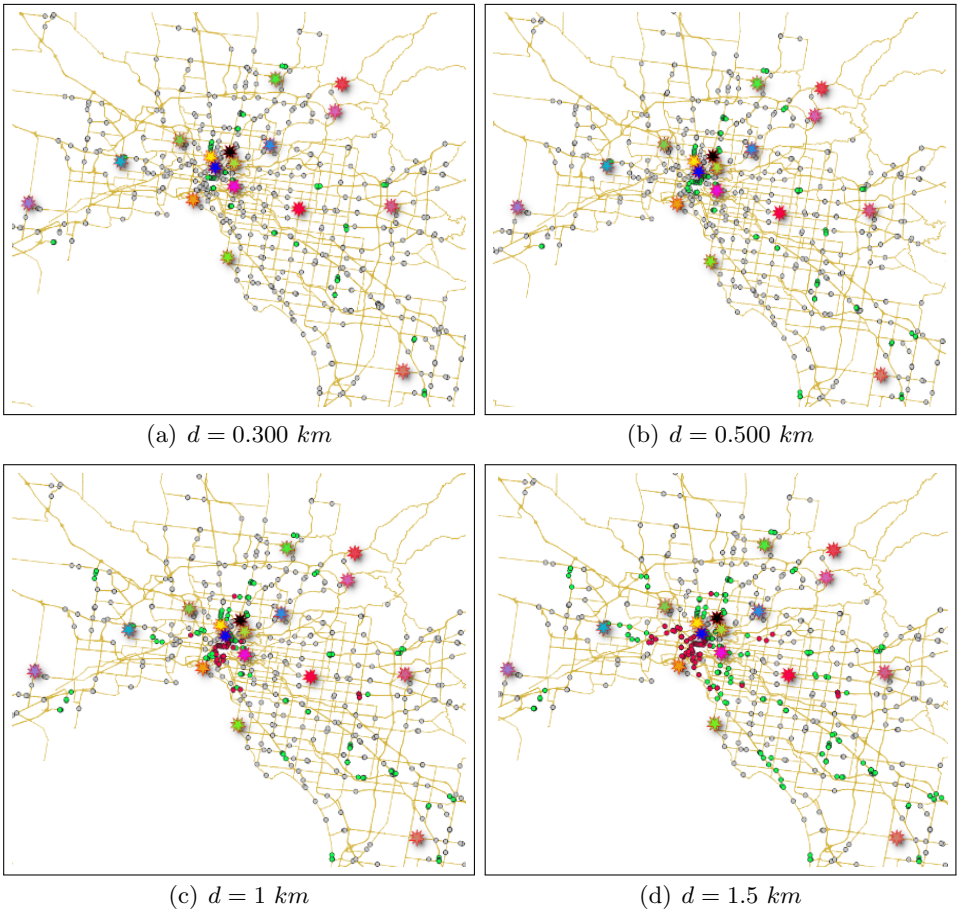
(a) Euclidean RNN query (b) Network RNN query

(c) Euclidean RNNH query (d) Network RNNH-RN query

Figure 5.17: Spatial query results on Map B

We can see that the results are different because Euclidean distance does not account for obstacles (such as a lake). Consequently, the result of reverse nearest neighbourhood (RNNH) when $d = 0.500\ km$ and $m = 4$, will also differ between the road network distance and the Euclidean distance as shown in Figure5.17(c) and Figure5.17(d).

In terms of the number of neighbourhoods obtained by queries, both Euclidean and network distance obtain two neighbourhoods when the $d = 0.500$ (circle and triangle neighbourhood), but the neighbourhood members are different. Also, we see that the number of neighbourhood (NH) members in each neighbourhood is different as shown in Figure 5.17.



(a) $d = 0.300\ km$ (b) $d = 0.500\ km$

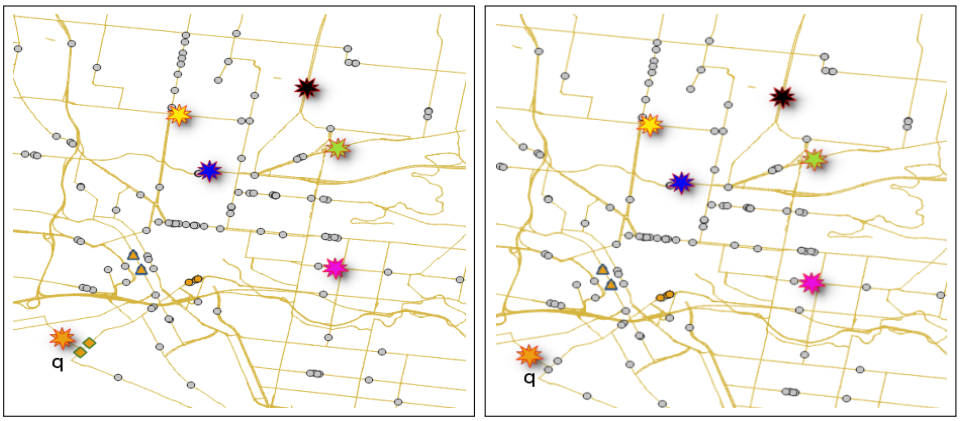
(c) $d = 1\ km$ (d) $d = 1.5\ km$

Figure 5.18: Green POI belongs only one NH, red POI belongs two NHs, Melbourne City

Figure 5.18 shows the flexibility of our proposed algorithm with different d values and the m value equals four. The red points are those belonging to two queries, the green points are those belonging to only one query and the grey points do not belong to any query. It is very clear that the RNNH-RN algorithm is not mutually exclusive; hence, the neighbourhood members can belong to two neighbourhoods for two queries as shown in Figure 5.18. Figure 5.18(a) and Figure 5.18(b) show the result of RNNH-RN when the $d = 0.300$ and $d = 0.500\ km$. The results of RNNH-RN show that for the points belonging to one neighbourhood, the number

of neighbourhood members increases when the d value increases. In Figures 5.18(c) and 5.18(d), the number of red points which belong to more than query increases when an increase in the d value also occurs.

In terms of minimum value in neighbourhood (m), we study the effect of changing the m value. When the value changes and increases from two to four, we find that the result excludes one neighbourhood based on the query preferences. The result of the orange query has three neighbourhoods: represented by a diamond, circle and triangle (two, four and six neighbourhood members respectively), when the m value equals two as shown Figure 5.19(a). On the other hand, Figure 5.19(b) shows a reverse nearest neighbourhood query when the m value equals four. We observe that it returns only two neighbourhoods (shaped as circle and triangle) but excludes the neighbourhoods that do not satisfy the minimum number of neighbourhood members. Figure 5.19 illustrates the effect of the m value on RNNH-RN queries.

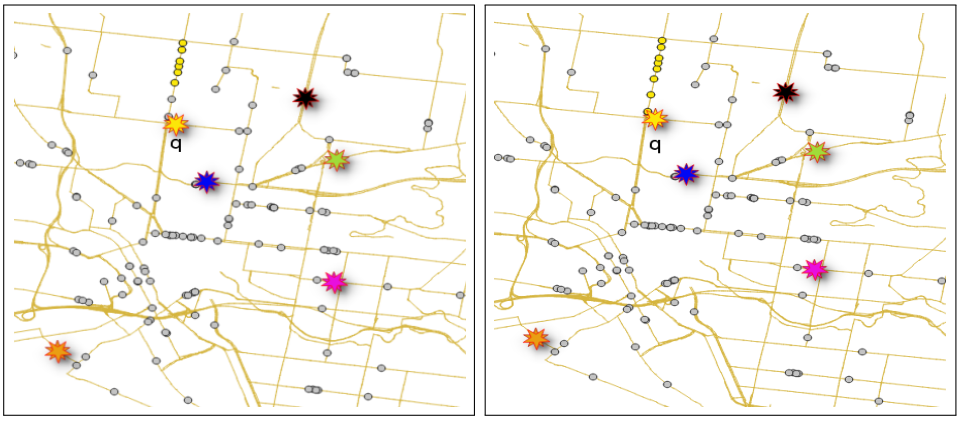


(a) $m = 2$ (b) $m = 4$

Figure 5.19: RNNH-RN result for $d = 0.5km$

As shown in Figure 5.20, the results change according to the value. Figure 5.20(a) depicts example of the reverse nearest neighbourhood on road networks (yellow query) when the $d = 0.300\ km$. It returns one neighbourhood with seven members located close to each other within 0.300 km. In Figure 5.20(b), when the maximum networks distance is 0.500 km, the yellow query involves the other two members that have a road network distance of less than 0.500 km and more than 0.300 km from the neighbourhood.

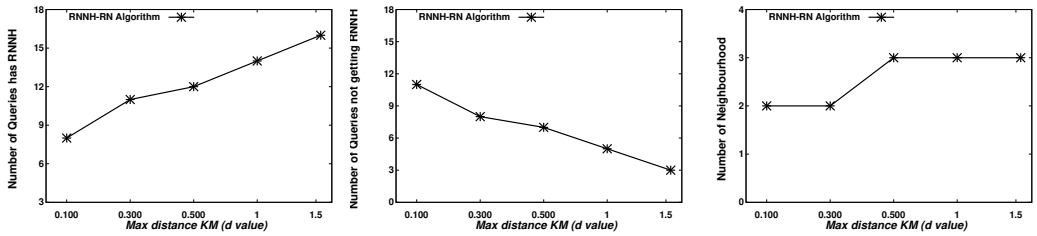
We report the results of reverse nearest neighbourhood on road networks based on the query points with different values of maximum distance (d) and minimum



(a) $d = 0.300$ (b) $d = 0.500$

Figure 5.20: RNNH-RN result for $m = 4$

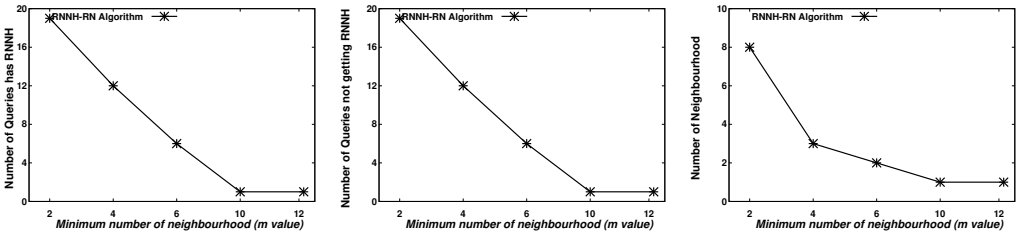
values in neighbourhood (m). The study illustrates how many queries could obtain a reverse nearest neighbourhood on road networks (RNNH-RN) with varying maximum distances between neighbourhood members in increments of 0.100, 0.300, 0.500, 1 and 1.50 km as depicted in Figure 5.21(a). Figure 5.21(b) shows how many queries do not obtain neighbourhoods; it is clear that when the maximum distance value increases, there is a stronger possibility of obtaining a neighbourhood.



(a) Number of queries with RNNH (b) Number of queries without RNNH (c) The average number of neighbourhoods

Figure 5.21: Variety of d values

We report the number of neighbourhoods obtained for each query with different values of maximum distance (d value), which increases when the maximum road distance increases. The average number of neighbourhoods increases to three when the maximum network distance reaches 0.500 km as indicated in Figure 5.21(c). The reason for this is that when the maximum distance (d) value increases, it in turn increases the chance of obtaining more points of interest in the neighbourhood. Therefore, there is a better chance of obtaining a better query result , when the maximum distance (d value) between NH members increases.



(a) Number of queries with RNNH (b) Number of Queries without RNNH (c) The average number of neighbourhoods

Figure 5.22: Variety of m values

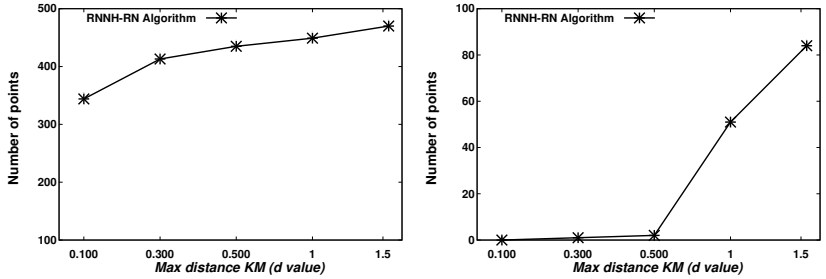
Our The experiment results led to the following conclusions. When the maximum distance (d) between neighbourhood members increases, the possibility of obtaining more neighbourhoods increases dramatically. Whether or not an RNNH-RN neighbourhood is obtained depends on the closeness of points of interest to each other. Moreover, the location of network Voronoi diagram of a query is limited by the location of other competitor facilities, which can limit the number of neighbourhoods obtained.

Conversely, when the value of m increases, the number of queries obtaining NH decreases dramatically as shown in Figure 5.22. Also, we find that the average number of neighbourhoods decreases, when the values of m increases. This is because there is a small group of points in the result set when m decreases.

We also find that the number of points belonging to one neighbourhood or more neighbourhoods and the average of number neighbourhood members increase dramatically when the value of d increases. However, the number of points not belonging to any neighbourhood decreases incrementally with an increase in the d value. This is because when the d value increases, it in turn increases the likelihood of having more points of interest in the neighbourhood as shown in Figure 5.23.

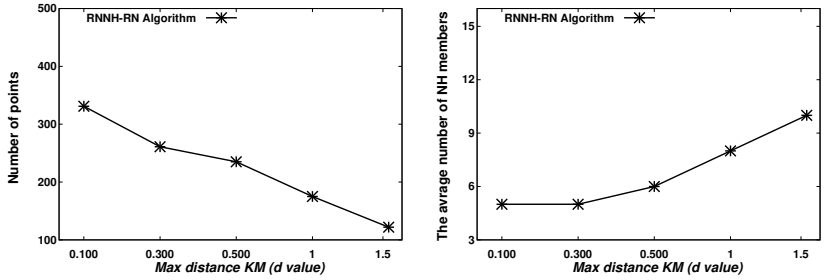
In our study, we examine the accuracy of results and compare the performance of the reverse nearest neighbourhood algorithm in two environments: road networks and Euclidean distance. We found that the number of neighbourhood members is much higher in Euclidean distance than in a road network as shown in Figure 5.24. This is because obstacles such as lakes, buildings, parks etc. are not taken into account when processing the neighbour-hood using Euclidean RNNH.

Also, we note that for both the Euclidean distance and the road network algorithms, the number of neighbourhood members decreases dramatically with an increase in m. However, the number of neighbourhood members increases immedi-



(a) The number of points belonging to only one neighbourhood

(b) The number of points belonging to more than two neighbourhoods

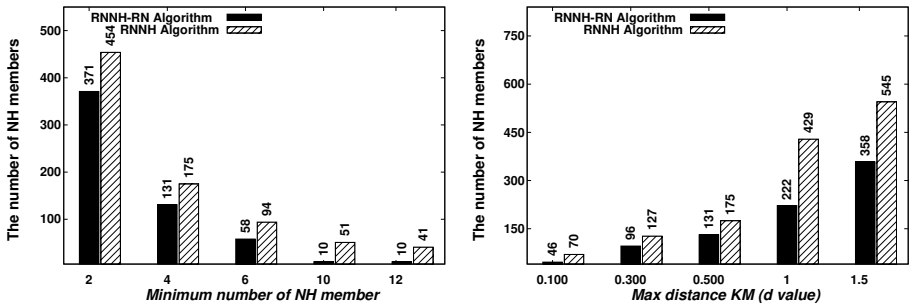


(c) The number of points not belonging to any neighbourhood

(d) The average number of neighbourhood members

Figure 5.23: Analysis for Melbourne City

ately with an increase in the maximum distance between neighbourhood members (d) for the reverse nearest neighbourhood in both road networks and Euclidean distance. We find that the efficiency of the RNNH-RN query processing algorithm in road networks improves when the d increase and the neighbourhood members on road networks is 9% fewer than for Euclidean RNNH.



(a) m value

(b) d value

Figure 5.24: Comparison RNNH and RNNH-RN algorithm for Melbourne City

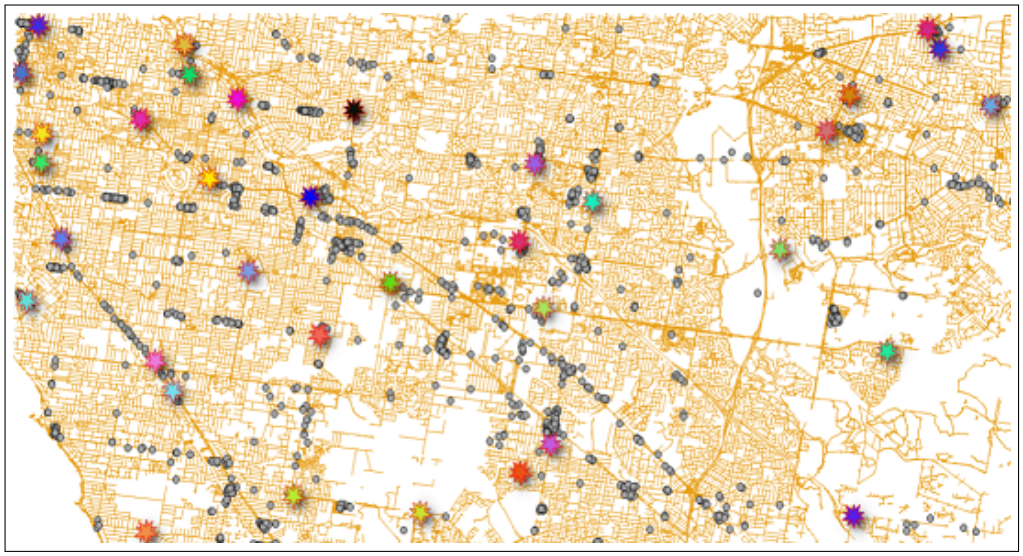


Figure 5.25: Stores located in South-East Melbourne, Victoria, Australia

5.5.3 High-density Experiment

In this experiment, we use the dataset for a high-density environment that includes a variety of stores in the South-East Melbourne area comprising cafés, restaurants and grocery stores. We choose an area that has a large number of stores in order to test the solution in a high-density environment. This enables us to test the feasibility of our algorithm and compare the results with the Euclidean RNNH algorithm in a high-density scenario.

The area of the South-East dataset is 370 km^2, and contains around 1,865 stores. The graph for this dataset has around 195160 vertices and 215400 edges. The stores represent points of interest, 3% of which are supplier facilities. We conducted a geospatial analysis to describe and understand the behaviour of the RNNH-RN algorithm in a high-density environment.

We conduct a geospatial analysis study of the nearest point in the graph, we calculate the network distance between each point of interest (POI) to the nearest neighbour. As shown in the graph the maximum network distance to the nearest neighbour is 5.13 km, while the average networks distance from one point to the nearest neighbour approximately is 100 m.

We also calculate the total distance travelled to the nearest point in the graph; it required 181 km to traverse the whole graph to reach the nearest point. Most points

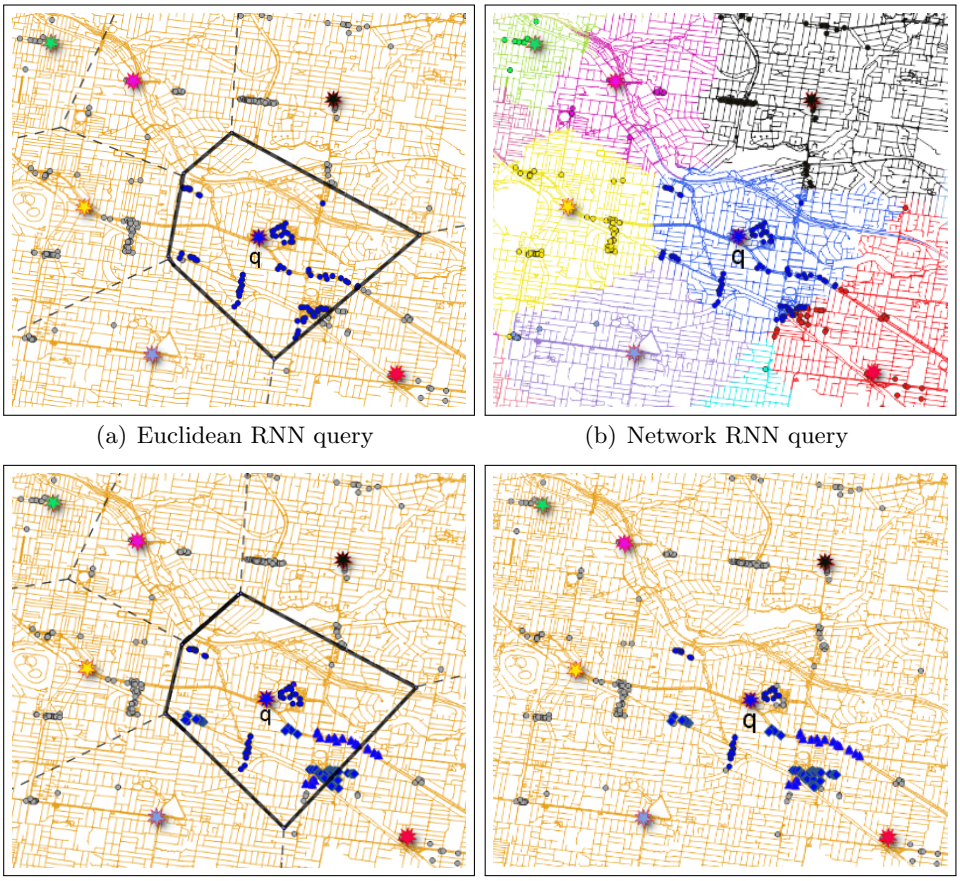
Table 5.6: Experimental analysis for Nearest Neighbour South-East Melbourne

Distance from	**Number of shops**
1-100 m	1458
100-200 m	208
200-300 m	61
300-400 m	29
400-800 m	44
800 m - 1 km	28
1-5 km	31
above 5 km	5
The maximum distance : 5.13 km **Average distance to nearest shop: 0.1021 km**	

are around 100 m from their nearest neighbour, while only a few points are more than 5 km from the nearest neighbour. The results of the analysis for the nearest neighbours are given in Table B.18 with details.

We select one random query, the blue query as an example, from the dataset to study the outcome of the difference of Reverse Nearest Neighbourhood in Euclidean distance and road networks, as shown in Figure 5.26. As seen in Figure 5.26(a) and Figure 5.26(b), the results of reverse nearest neighbour (RNN) in Euclidean and road networks distance for selected queries are different, where it is clear that the outcome of the RNN in Euclidean distance involves more points in the result set than does the road network distance. Also, the outcome obtained by the Voronoi diagram is different. This is because the Euclidean distance depends on the relative positions of the two points, while the distance on a road network depends on the relative positions as well as the sections of road between the two points.

Consequently, this has an impact on the outcome of the reverse nearest neighbourhood query in terms of distance in Euclidean and road networks as shown in Figure 5.26(c) and Figure 5.26(d). When the set has values of $d = 0.200$ km and $m = 4$ the Euclidean RNNH returns the same number of neighbourhoods as does the Reverse Nearest Neighbourhood on road networks (RNNH-RN), although it involves more points because the road network distance is not taken into account. For instance, the first neighbourhood, has more neighbourhood members in terms of the



(a) Euclidean RNN query (b) Network RNN query

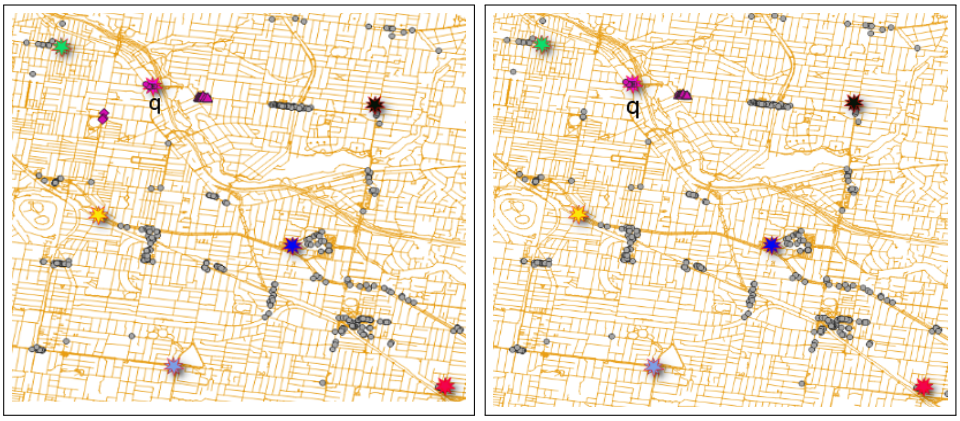
(c) Euclidean RNNH query (d) Network RNNH-RN query

Figure 5.26: Spatial query results on Map C

Euclidean distance because these points are less than 0.200 km Euclidean distance. Conversely, in the road network, these points have more than 0.200 km road distance.

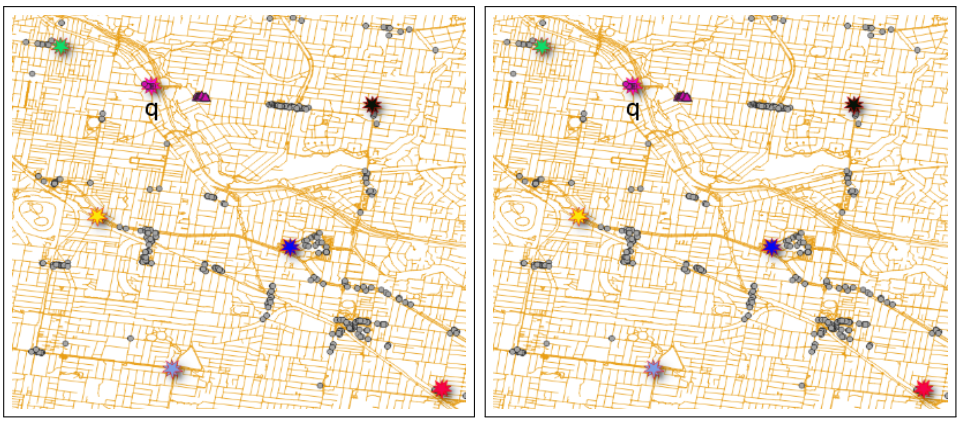
In regard to the minimum number of neighbourhood members (m), we take the purple query in Figure 5.27 as example. When the value of m changes and increases from two neighbourhoods to six neighbourhoods at minimum value, one neighbourhood is excluded based on the query preferences. In Figure 5.27(a), the minimum value of neighbourhood members is two, then the query returns three neighbourhoods. Figure 5.27(a) shows the result when the minimum number of neighbourhood members is six. It returns only two neighbourhoods that have at least six neighbourhood members. In this example, there is only one neighbourhood.

Figure 5.28 shows the results of varying the value of the maximum network distance (d). Figure 5.28(a) depicts an example of the RNNH query (purple query) where $d = 100\ m$. The same number of neighbourhoods is obtained even though



(a) $m = 2$ (b) $m = 6$

Figure 5.27: RNNH-RN result for $d = 0.200km$

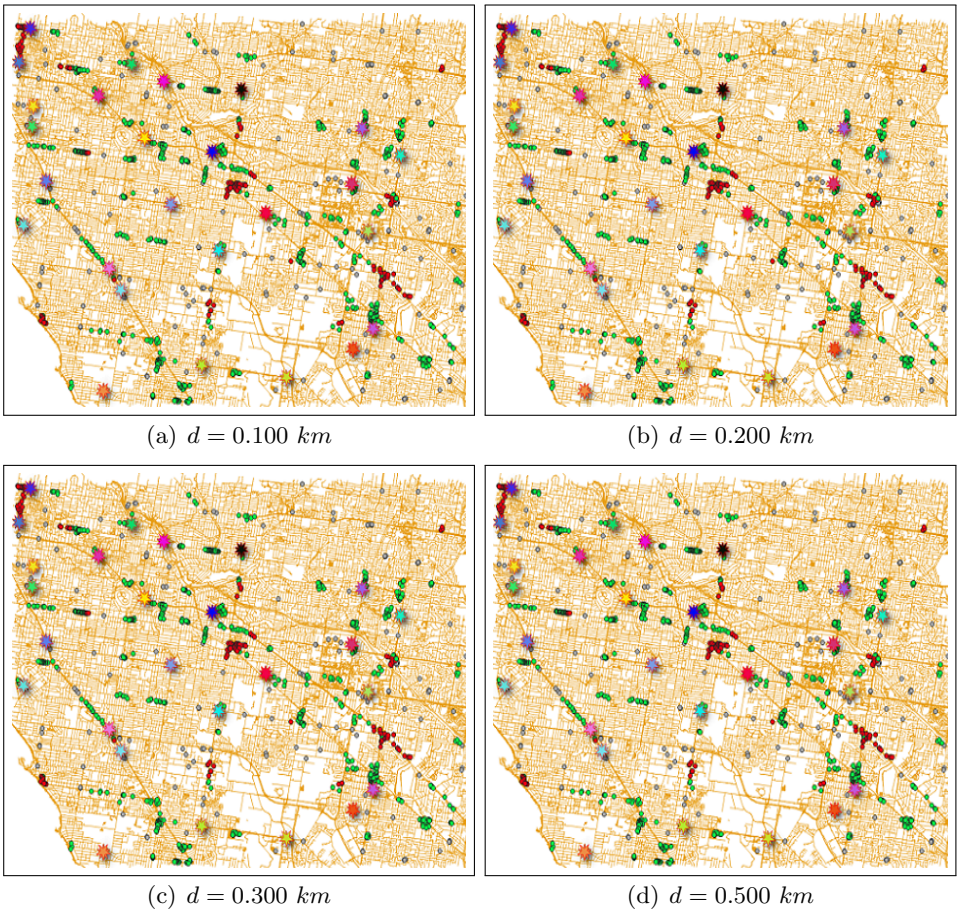


(a) $d = 0.100$ (b) $d = 0.500$

Figure 5.28: RNNH-RN result for $m = 6$

the maximum network distances are different as shown in Figure 5.28(b). The same number of neighbourhoods are returned when the maximum network distance is 500 m. This is because in a high-density dataset, the points of interest are located very close to each other, so the neighbourhood members are within a short range of each other.

Also, we study the behaviour of the algorithm using different values for the maximum network distance (d) and a static value for the minimum number of neighbourhood members ($m = 4$). We study the flexibility of our proposed algorithm and note how many points are not mutually exclusive in a high-density dataset. The red points are the points belonging to more than one query, the green points are the points belonging to only one query, the remaining grey points do not belong to any



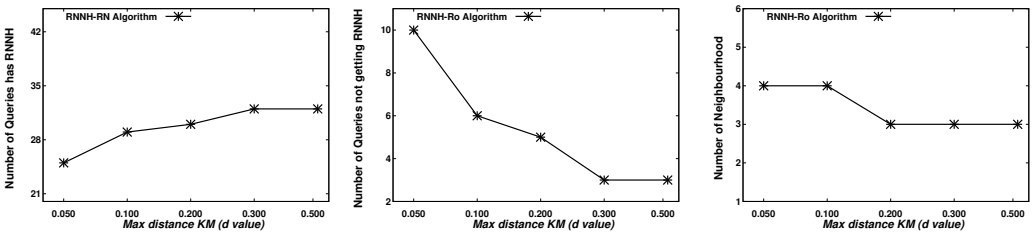
(a) $d = 0.100\ km$ (b) $d = 0.200\ km$

(c) $d = 0.300\ km$ (d) $d = 0.500\ km$

Figure 5.29: Green POI belongs only one NH, red POI belongs two NHs, South East area

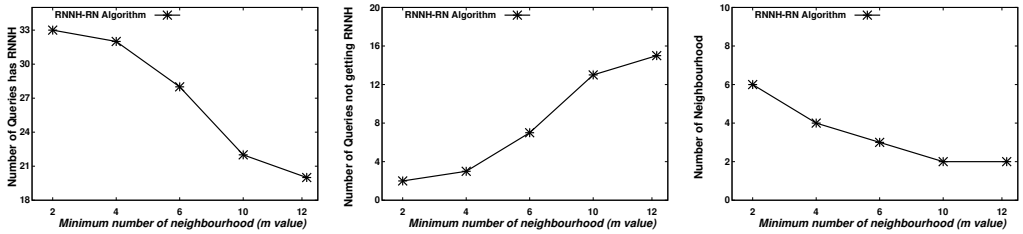
query. We see that the number of points of interest belong to only one neighbourhood (NH) and those in more than one neighbourhood (NH) remain stable even when the value of d increases by five times from 0.100 km to 0.500 km as depicted in Figure 5.29. This is because the query is distributed and located very close to the cluster of points of interest (POIs).

In addition, as the density increases, the number of queries required to obtain RNNH-RN increases as well as the value of d increases. We study the effect of the d value on the efficiency of RNNH-RN. This study illustrates the number of queries required to obtain RNNH-RN and the average number of neighbourhoods obtained by each query with different values of d. We conclude that the possibility of obtaining reverse nearest neighbourhoods increases dramatically, when the maximum distance (d) between neighbourhood members increases. Second, we found that the number of neighbourhoods for each query declines with an increase in the maximum distance



(a) Number of queries with RNNH (b) Number of queries without RNNH (c) The average number of neighbourhoods

Figure 5.30: Variety of d values



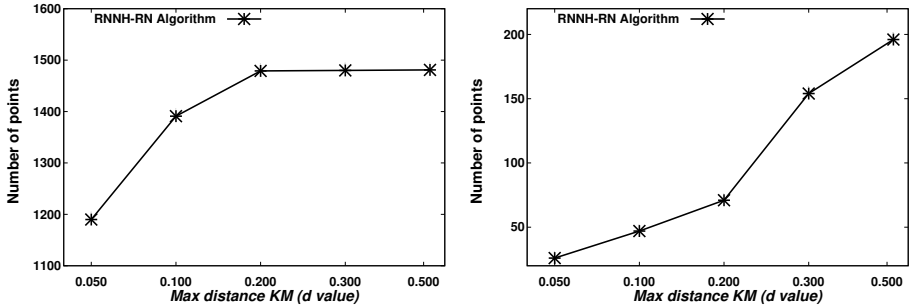
(a) Number of queries with RNNH (b) Number of Queries without RNNH (c) The average number of neighbourhoods

Figure 5.31: Variety of m values

between neighbourhood members (d value). When d increases, so does the number of neighbourhood members, since they are combined together into one neighbourhood (NH). Figure 5.30 shows the results of varying the d value.

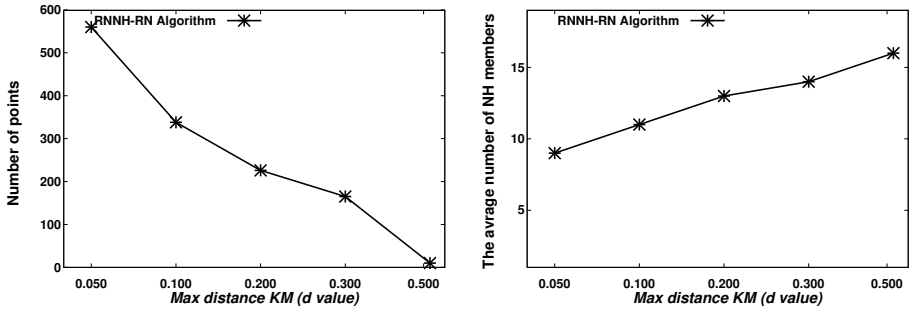
Figure 5.31 illustrates the number of queries required to obtain RNNH-RN and the average number of neighbourhoods obtained by each query with different values for the minimum number of neighbourhood members (m). As we have seen when the value of the minimum number of neighbourhood members increases, the number of queries required to obtain neighbourhood (NH) and the average number of neighbourhoods decreases dramatically regardless of the environment. Also, we conclude that when the value of (m) increases, it reduces the likelihood of having more neighbourhoods. This excludes, some neighbourhoods but does not satisfy the criterion regarding maximum network distance value (d) .

Also, we find that the number of points belonging to one or more neighbourhoods and the average number of neighbourhood members increase dramatically when the maximum network distance between neighbourhood members (d value) is increased. Conversely the number of points that do not belong to any neighbourhood decreases incrementally with an increase in the maximum network distance (d). The reason is



(a) The number of points belonging to only one neighbourhood

(b) The number of points belonging to more than two neighbourhoods



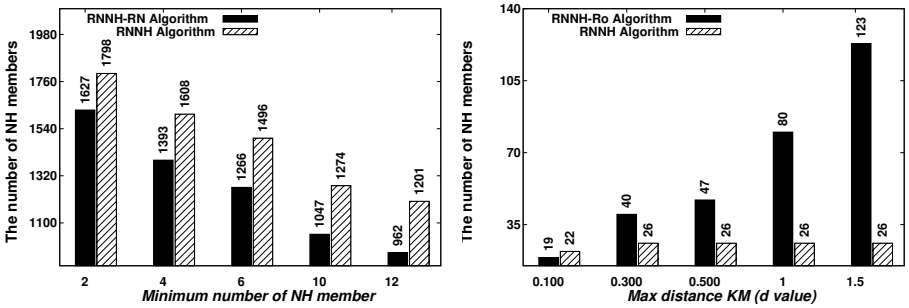
(c) The number of points not belonging to any neighbourhood

(d) The average number of neighbourhood members

Figure 5.32: Analysis for South-East Melbourne

that, as shown in Figure 5.32, when the maximum distance (d) value increases, this increases the likelihood of having more points of interest in the neighbourhood.

We have compared Reverse Nearest Neighbourhood in road networks and Euclidean distance algorithm as shown in Figure 5.33. We conclude that in the Euclidean Reverse Nearest neighbourhood (RNNH) algorithm the number of neighbourhood members is much higher than for the Reverse Nearest Neighbourhood in road network (RNNH-RN) algorithm in all three experiments. Also, in both algorithms, the number of neighbourhood members increases immediately with an increase in the maximum distance between neighbourhood members (d). Furthermore, it decreases dramatically with an increase in the minimum number of neighbourhood members (m value) in Euclidean and road networks. This is because when the maximum distance (d) increases, there is a greater chance of having more neighbourhood members, which leads to greater search distance and includes these members in the result set. We examine the accuracy of our proposed algorithm in road networks, and find that the neighbourhood members on road networks is 30% fewer than for Euclidean RNNH as the road network reflects actual distances in all scenarios.



(a) m value (b) d value

Figure 5.33: Comparison RNNH and RNNH-RN algorithm for South-East Melbourne

5.6 Conclusion

In this chapter, we studied the problem of Reverse Nearest Neighborhood in Road Networks. Reverse Nearest Neighborhood in road networks (RNNH-RN) queries are motivated by our observation that Reverse Nearest Neighbor queries may be unable to properly capture the notion of influence points. Hence, we introduced the concept of a neighborhood version of reverse nearest neighbour as a collection of chained points that are located along road networks. Our proposed approach is intended to find the most accessible neighbourhood that considers the query as the nearest facility on road networks while maintaining all its formal properties (not mutual exclusiveness). We found that, compared to Euclidean RNNH (chapter 4), the amount of reduction for neighbourhood members in RNNH on road networks is twenty percent on average. Extensive experiments were conducted to present a thorough theoretical analysis of a spatial network and demonstrate a viable solution for Reverse Nearest Neighborhood in Road Networks queries, which is applicable to different densities of datasets.

Chapter 6

Conclusion and Future Works

6.1 Overview

In this thesis, we have presented efficient techniques for Reverse Nearest Neighbourhood queries based on Euclidean and network distances under different settings. This chapter summarises the overall contribution of the thesis and offers suggestions for future research on this topic.

6.2 Conclusion

Nowadays, the grouping of spatial data is important because it is essential process in exploratory spatial data mining. It is a task involving the grouping of a set of objects in such a way that those in the same group are more similar (in some sense or another) to each other than to those in other groups. It is used widely by various applications. Many of our online activities today rely on spatial data such as Facebook and LinkedIn websites, and they have more than 600 million active users, and these users have chain connection Friends. At least half of these users log in every day and they actively access information through enabled location-based service (LBS) [172]. Therefore, understanding these spatial users is crucially important for the location-based service providers. The grouping approach is a successful means of achieving a better understanding of data.

In this thesis, we conducted research and proposed solutions for overcoming the problems of grouping data in the context of a chained group of objects in reverse nearest neighbour (RNN) queries. This study is the first to propose a new type of query, named the Reverse Nearest Neighbourhood Query, one that aims to find the most influential neighbourhoods for a given query. We have also introduced the first solution that aims to improve the efficiency of the monitoring of continuous reverse

nearest neighbourhood queries. We have verified the effectiveness of these proposed approaches with extensive empirical evaluations of synthetic and real-world data sets on road network and Euclidean distances. In this context, this thesis makes important contributions as detailed below.

In Chapter 3, we propose a new definition of influenced neighbourhood. In our definition, the neighbourhood has a number of points of interest where point p is said to be influenced by not only its closest facility but also other facilities that are very close to p. We are the first to introduce the Reverse Nearest Neighbourhood (RNNH) query to compute the chained neighbourhood for a given query. The clustering and grouping of points in our definition of reverse nearest neighbourhood is a unique clustering approach which has never been attempted previously in spatial databases. Our approach makes it possible for neighbourhood members to be influenced by the query point and not be considered as sparse objects. Furthermore, the neighbourhood considers the query point as the nearest facility among all the facilities. All of the above features are unique to our proposed algorithm compared with existing algorithms in spatial databases such as [13]. We have shown that all existing RNN algorithms cannot be applied for computing the neighbourhood with the above the features. We therefore propose more efficient and pruning techniques and use them in our RNNH algorithm. In our experimental study that using real and synthetic data sets, we prove that our algorithm is very efficient when compared to the naïve algorithm or traditional spatial queries. This work was published in [22].

In Chapter 4, we extend our work on static (RNNH) queries for continuous RNNH queries. The main major challenges for continuous query is monitoring the query in an efficient way. Consequently, in our study we focused on two main goals: minimising the frequent updates of the query location; and keeping the neighbourhood results unchanged while monitoring the moving query. We are the first to propose a neighbourhood safe region method for monitoring continuous Reverse Nearest Neighbourhood (RNNH) queries. We demonstrated the uniqueness of our method as one that avoids supplementary communication between query and server while the query is in the safe region. This significantly reduces the need for monitoring continuous (RNNH) query while it is inside the safe region. Also, it eliminates the need for a query to follow a defined path while it is in the safe region. We have conducted evaluations on the performance of the neighbourhood safe region method. Results demonstrate that our safe region method can effectively monitor (RNNH) queries because it reduces computation and communication costs while at the same time monitoring continuous queries. Our technique of monitoring the moving query was published in [23].

In Chapter 5, we present a new framework to capture the notion of neighbourhood on spatial road networks. Ours is the first to study the Reverse Nearest Neighbourhood on road networks. The technique and problem definition investigated in Chapter 3 is not applicable to a road network environment, calculate the distance in a road network environment is more complicated. Thus, we propose new efficient techniques for computing the neighbourhood on road networks, which we named Reverse Nearest Neighbourhood on Road networks (RNNHRO) queries. In our study, we explain the unique definition of neighbourhood in the road network setting. Our experimental study shows that our algorithm is significantly feasible when applied to spatial road network.

Our proposed framework known as " Reverse Nearest Neighbourhood search in spatial databases", can achieve a better and more meaningful clustering of reverse nearest neighbours. It is anticipated this research project will significantly benefit and add to the continuing improvement of data clustering in spatial databases.

6.3 Future Work

There are several promising directions for future research where our idea could be extended to include other areas. This section outlines the possible directions for future works.

6.3.1 Reverse Spatial Top-k Neighbourhood Query

In Chapter 3, we present reverse nearest Neighbourhood queries and the computation of such queries when distance is the only factor being considered. In many real world scenarios, users may consider other factors in choosing a facility, such as price, rating, etc [173, 174, 175]. Hence, it will be interesting to study the reverse nearest neighbourhood top-k queries where the objective is to consider other factors besides distance. In other words, the reverse spatial top-k Neighbourhood queries return every neighbourhood for which the score of the neighbourhood is almost as good as the most preferred for given query. Also, we can consider multiple factors in RNNH methods to find the answer for query.

6.3.2 Improving the Effectiveness of Safe Region for Reverse Nearest Neighbourhood Query

In Chapter 4, we propose a safe region which is an area where the query does not need to send its location to the server as long as it is inside its safe region. As stated in Chapter 4, we assume that only the query is moving. We are interested in exploring a scenario where the users are moving; it will be a challenge for such a study to provide a framework for the continuous monitoring of moving users and facilities. In fact, it will be very interesting to analyse the trade-off between computation cost and communication cost.

6.3.3 Reverse Nearest Neighbourhood on Road Network $RNNH - RO$ Query for Moving Queries

In Chapter 5, we propose Reverse Nearest Neighbourhood on Road Network $RNNH-RO$ Query. Our technique and algorithm are able to solve the Reverse nearest neighbourhood on road networks for a stationary query. Possible future research could investigate the $RNNH - RO$ queries in road network for continuous query. Many applications can be used in this scenario and such as cab/taxi dispatching system. In case of a big event held, developing continuous RNNH algorithm in the road network, will help to know how many cabs can be around the event and it will reduce the waiting time of the cab drivers to get customers. The problem definition as noted in Chapter 5 is relevant in road network settings, but it is not applicable for continuous query. Moreover, we are interested in including the spatial keyword queries for computing reverse nearest neighbourhood on road networks.

Bibliography

[1] Antonin Guttman. *R-trees: a dynamic index structure for spatial searching*, volume 14. ACM, 1984.

[2] Muhammad Aamir Cheema. Circulartrip and arctrip: Effective grid access methods for continuous spatial queries. Master's thesis, School of Computer Science and Engineering, The University of New South Wales, Sydney Australia, 3 2007.

[3] Muhammad Aamir Cheema, Xuemin Lin, Wenjie Zhang, and Ying Zhang. Influence zone: Efficiently processing reverse k nearest neighbors queries. In *ICDE*, pages 577–588, 2011.

[4] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

[5] David Taniar and Wenny Rahayu. A taxonomy for nearest neighbour queries in spatial databases. *J. Comput. Syst. Sci.*, 79(7):1017–1039, 2013.

[6] Thao P Nghiem, Kiki Maulana, Kinh Nguyen, David Green, Agustinus Borgy Waluyo, and David Taniar. Peer-to-peer bichromatic reverse nearest neighbours in mobile ad-hoc networks. *Journal of Parallel and Distributed Computing*, 74(11):3128–3140, 2014.

[7] Omar Al-Bayari and Balqies Sadoun. New centralized automatic vehicle location communications software system under gis environment. *Int. J. Communication Systems*, 18(9):833–846, 2005.

[8] Chongsheng Zhang, George Almpanidis, Faegheh Hasibi, and Gaojuan Fan. Gridvoronoi: An efficient spatial index for nearest neighbor query processing. *IEEE Access*, 7:120997–121014, 2019.

[9] Dimitris Papadias, Qiongmao Shen, Yufei Tao, and Kyriakos Mouratidis. Group nearest neighbor queries. In *Proceedings. 20th International Conference on Data Engineering*, pages 301–312. IEEE, 2004.

[10] Kiki Maulana Adhinugraha, David Taniar, and Maria Indrawan. Finding reverse nearest neighbors by region. *Concurrency and Computation: Practice and Experience*, 26(5):1142–1156, 2014.

[11] Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[12] Maytham Safar, Dariush Ibrahimi, and David Taniar. Voronoi-based reverse nearest neighbor query processing on spatial networks. *Multimedia systems*, 15(5):295–308, 2009.

[13] Anasthasia Agnes Haryanto, David Taniar, and Kiki Maulana Adhinugraha. Group reverse knn query optimisation. *Journal of Computational Science*, 11:205–221, 2015.

[14] Sultan Alamri, David Taniar, and Maytham Safar. A taxonomy for moving object queries in spatial databases. *Future Generation Comp. Syst.*, 37:232–242, 2014.

[15] Muhammad Aamir Cheema, Xuemin Lin, Wenjie Zhang, and Ying Zhang. A safe zone based approach for monitoring moving skyline queries. In *EDBT*, pages 275–286. ACM, 2013.

[16] Katie Elizabeth Potts, Rohan Mark Bennett, and Abbas Rajabifard. Spatially enabled bushfire recovery. *GeoJournal*, 78(1):151–163, 2013.

[17] Raphaele Blanchi, Justin Leonard, Katharine Haynes, Kimberley Opie, Melissa James, and Felipe Dimer de Oliveira. Environmental circumstances surrounding bushfire fatalities in australia 1901–2011. *Environmental Science & Policy*, 37:192–203, 2014.

[18] Muhammad Aamir Cheema, Ljiljana Brankovic, Xuemin Lin, Wenjie Zhang, and Wei Wang. Multi-guarded safe zone: An effective technique to monitor moving circular range queries. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 189–200. IEEE, 2010.

[19] Sarana Nutanong, Rui Zhang, Egemen Tanin, and Lars Kulik. The v*-diagram: a query-dependent approach to moving knn queries. *Proceedings of the VLDB Endowment*, 1(1):1095–1106, 2008.

[20] Mohammad Kolahdouzan and Cyrus Shahabi. Voronoi-based k nearest neighbor search for spatial network databases. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 840–851, 2004.

[21] Cyrus Shahabi, Mohammad R Kolahdouzan, and Mehdi Sharifzadeh. A road network embedding technique for k-nearest neighbor search in moving object databases. *GeoInformatica*, 7(3):255–273, 2003.

[22] Nasser Allheeib, Md Saiful Islam, David Taniar, Zhou Shao, and Muhammad Aamir Cheema. Density-based reverse nearest neighbourhood search in spatial databases. *J. Ambient Intelligence and Humanized Computing*, pages 1–12, 2018.

[23] Nasser Allheeib, David Taniar, Haidar Al-Khalidi, Md Saiful Islam, and Kiki Maulana Adhinugraha. Safe regions for moving reverse neighbourhood queries in a peer-to-peer environment. *IEEE Access*, 8:50285–50298, 2020.

[24] Muhammad Aamir Cheema, Xuemin Lin, Wenjie Zhang, and Ying Zhang. Influence zone: Efficiently processing reverse k nearest neighbors queries. In *ICDE*, pages 577–588. IEEE, 2011.

[25] Dimitris Papadias, Jun Zhang, Nikos Mamoulis, and Yufei Tao. Query processing in spatial network databases. In *VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany*, pages 802–813, 2003.

[26] Dongsheng Li, Jiannong Cao, Xicheng Lu, and Kaixian Chen. Efficient range query processing in peer-to-peer systems. *IEEE Trans. Knowl. Data Eng.*, 21(1):78–91, 2009.

[27] Apostolos Papadopoulos and Yannis Manolopoulos. Multiple range query optimization in spatial databases. In *Advances in Databases and Information Systems, Second East European Symposium, ADBIS'98, Poznan, Poland, Spetember 7-10, 1998, Proceedings*, pages 71–82, 1998.

[28] Jing Shan, Donghui Zhang, and Betty Salzberg. On spatial-range closest-pair query. In *Advances in Spatial and Temporal Databases, 8th International Symposium, SSTD 2003, Santorini Island, Greece, July 24-27, 2003, Proceedings*, pages 252–269, 2003.

[29] Hoong Kee Ng and Hon Wai Leong. Path-based range query processing using sorted path and rectangle intersection approach. In *Database Systems for Advances Applications, 9th International Conference, DASFAA 2004, Jeju Island, Korea, March 17-19, 2004, Proceedings*, pages 184–189, 2004.

[30] Muhammad Aamir Cheema, Ljiljana Brankovic, Xuemin Lin, Wenjie Zhang, and Wei Wang. Continuous monitoring of distance-based range queries. *IEEE Trans. Knowl. Data Eng.*, 23(8):1182–1199, 2011.

[31] Shyh-Kwei Chen, Kun-Lung Wu, and Philip Shi-lung Yu. Range query methods and apparatus, September 24 2013. US Patent 8,543,579.

[32] Dimitris Papadias, Jun Zhang, Nikos Mamoulis, and Yufei Tao. Query processing in spatial network databases. In *Proceedings 2003 VLDB Conference*, pages 802–813. Elsevier, 2003.

[33] Nasser Ghadiri, Ahmad Baraani-Dastjerdi, Nasser Ghasem-Aghaee, and Mohammad A Nematbakhsh. Optimizing the performance and robustness of type-2 fuzzy group nearest-neighbor queries. *Mobile Information Systems*, 7(2):123–145, 2011.

[34] Quoc Thai Tran, David Taniar, and Maytham Safar. Reverse k nearest neighbor and reverse farthest neighbor search on spatial networks. In *Transactions on large-scale data-and knowledge-centered systems I*, pages 353–372. Springer, 2009.

[35] Kefeng Xuan, Geng Zhao, David Taniar, Wenny Rahayu, Maytham Safar, and Bala Srinivasan. Voronoi-based range and continuous range query processing in mobile databases. *J. Comput. Syst. Sci.*, 77(4):637–651, 2011.

[36] Michiel Smid. Closest-point problems in computational geometry. In *Handbook of computational geometry*, pages 877–935. Elsevier, 2000.

[37] Nick Roussopoulos, Stephen Kelley, and Frédéric Vincent. Nearest neighbor queries. In *ACM Sigmod Record*, volume 24, pages 71–79. ACM, 1995.

[38] Gísli R Hjaltason and Hanan Samet. Distance browsing in spatial databases. *ACM Transactions on Database Systems (TODS)*, 24(2):265–318, 1999.

[39] Nick Roussopoulos, Stephen Kelley, and Frédéic Vincent. Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 22-25, 1995.*, pages 71–79, 1995.

[40] Gísli R. Hjaltason and Hanan Samet. Distance browsing in spatial databases. *ACM Trans. Database Syst.*, 24(2):265–318, 1999.

[41] Thomas Seidl and Hans-Peter Kriegel. Optimal multi-step k-nearest neighbor search. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA.*, pages 154–165, 1998.

[42] Christian S. Jensen, Jan Kolárvr, Torben Bach Pedersen, and Igor Timko. Nearest neighbor queries in road networks. In *GIS*, pages 1–8, 2003.

[43] M. Kolahdouzan and Cyrus Shahabi. Voronoi-based k nearest neighbor search for spatial network databases. In *VLDB*, pages 840–851, 2004.

[44] Mohammad R. Kolahdouzan and Cyrus Shahabi. Continuous k-nearest neighbor queries in spatial network databases. In *Spatio-Temporal Database Management, 2nd International Workshop STDBM'04, Toronto, Canada, August 30, 2004*, pages 33–40, 2004.

[45] Cyrus Shahabi, Mohammad R. Kolahdouzan, and Mehdi Sharifzadeh. A road network embedding technique for k-nearest neighbor search in moving object databases. In *ACM-GIS*, pages 94–10, 2002.

[46] Hyung-Ju Cho and Chin-Wan Chung. An efficient and scalable approach to cnn queries in a road network. In *VLDB*, pages 865–876, 2005.

[47] Haibo Hu, Dik Lun Lee, and Jianliang Xu. Fast nearest neighbor search on road networks. In *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006, Proceedings*, pages 186–203, 2006.

[48] Kyriakos Mouratidis, Man Lung Yiu, Dimitris Papadias, and Nikos Mamoulis. Continuous nearest neighbor monitoring in road networks. In *VLDB*, pages 43–54, 2006.

[49] Hua Lu, Xin Cao, and Christian S. Jensen. A foundation for efficient indoor distance-aware query processing. In *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*, pages 438–449, 2012.

[50] Xike Xie, Hua Lu, and Torben Bach Pedersen. Efficient distance-aware query evaluation on indoor moving objects. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pages 434–445, 2013.

[51] Jiao Yu, Wei-Shinn Ku, Min-Te Sun, and Hua Lu. An RFID and particle filter-based indoor spatial query evaluation system. In *Joint 2013 EDBT/ICDT Conferences, EDBT '13 Proceedings, Genoa, Italy, March 18-22, 2013*, pages 263–274, 2013.

[52] Joon-Seok Kim and Ki-Joune Li. Location k-anonymity in indoor spaces. *GeoInformatica*, 20(3):415–451, 2016.

[53] Jun Zhang, Dimitris Papadias, Kyriakos Mouratidis, and Manli Zhu. Query processing in spatial databases containing obstacles. *International Journal of Geographical Information Science*, 19(10):1091–1111, 2005.

[54] Chenyi Xia, David Hsu, and Anthony K. H. Tung. A fast filter for obstructed nearest neighbor queries. In *Key Technologies for Data Management, 21st British National Conference on Databases, BNCOD 21, Edinburgh, UK, July 7-9, 2004, Proceedings*, pages 203–215, 2004.

[55] Sarana Nutanong, Egemen Tanin, and Rui Zhang. Visible nearest neighbor queries. In *Advances in Databases: Concepts, Systems and Applications, 12th International Conference on Database Systems for Advanced Applications, DASFAA 2007, Bangkok, Thailand, April 9-12, 2007, Proceedings*, pages 876–883, 2007.

[56] Yunjun Gao, Baihua Zheng, Wang-Chien Lee, and Gencai Chen. Continuous visible nearest neighbor queries. In *EDBT 2009, 12th International Conference on Extending Database Technology, Saint Petersburg, Russia, March 24-26, 2009, Proceedings*, pages 144–155, 2009.

[57] Ze Deng, Meng Wang, Lizhe Wang, Xiaohui Huang, Wei Han, Junde Chu, and Albert Y Zomaya. An efficient indexing approach for continuous spatial approximate keyword queries over geo-textual streaming data. *ISPRS International Journal of Geo-Information*, 8(2):57, 2019.

[58] Andreas Henrich. A distance scan algorithm for spatial access structures. In *ACM-GIS*, pages 136–143, 1994.

[59] Flip Korn, Nikolaos Sidiropoulos, Christos Faloutsos, Eliot L. Siegel, and Zenon Protopapas. Fast nearest neighbor search in medical image databases. In *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 215–226, 1996.

[60] Apostolos Papadopoulos and Yannis Manolopoulos. Performance of nearest neighbor queries in r-trees. In *Database Theory - ICDT '97, 6th International Conference, Delphi, Greece, January 8-10, 1997, Proceedings*, pages 394–408, 1997.

[61] Norio Katayama and Shin'ichi Satoh. The sr-tree: An index structure for high-dimensional nearest neighbor queries. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA.*, pages 369–380, 1997.

[62] Hanan Samet, Jagan Sankaranarayanan, and Houman Alborzi. Scalable network distance browsing in spatial databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 43–54, 2008.

[63] Surajit Chaudhuri and Luis Gravano. Evaluating top-k selection queries. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 397–410, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[64] Chaluka Salgado, Muhammad Aamir Cheema, and Mohammed Eunus Ali. Continuous monitoring of range spatial keyword query over moving objects. *World Wide Web*, 21(3):687–712, 2018.

[65] Tinghuai Ma, Jing Jia, Yu Xue, Yuan Tian, Abdullah Al-Dhelaan, and Mznah Al-Rodhaan. Protection of location privacy for moving knn queries in social networks. *Applied Soft Computing*, 66:525–532, 2018.

[66] A. Prasad Sistla, Ouri Wolfson, Sam Chamberlain, and Son Dao. Modeling and querying moving objects. In *Proceedings of the Thirteenth International Conference on Data Engineering, April 7-11, 1997 Birmingham U.K.*, pages 422–432, 1997.

[67] George Kollios, Dimitrios Gunopulos, and Vassilis J. Tsotras. Nearest neighbor queries in a mobile environment. In *Spatio-Temporal Database Management, International Workshop STDBM'99, Edinburgh, Scotland, September 10-11, 1999, Proceedings*, pages 119–134, 1999.

[68] Zhexuan Song and Nick Roussopoulos. K-nearest neighbor search for moving query point. In *SSTD*, pages 79–96, 2001.

[69] Xiaopeng Xiong, Mohamed F. Mokbel, Walid G. Aref, Susanne E. Hambrusch, and Sunil Prabhakar. Scalable spatio-temporal continuous query processing for location-aware services. In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM 2004), 21-23 June 2004, Santorini Island, Greece*, pages 317–326, 2004.

[70] Yufei Tao and Dimitris Papadias. Time-parameterized queries in spatio-temporal databases. In *SIGMOD Conference*, pages 334–345, 2002.

[71] Simonas Saltenis, Christian S. Jensen, Scott T. Leutenegger, and Mario A. López. Indexing the positions of continuously moving objects. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, pages 331–342, 2000.

[72] Yufei Tao, Dimitris Papadias, and Jimeng Sun. The tpr*-tree: An optimized spatio-temporal access method for predictive queries. In *VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany*, pages 790–801, 2003.

[73] Rimantas Benetis, Christian S. Jensen, Gytis Karciauskas, and Simonas Saltenis. Nearest and reverse nearest neighbor queries for moving objects. *VLDB J.*, 15(3):229–249, 2006.

[74] Katerina Raptopoulou, Apostolos Papadopoulos, and Yannis Manolopoulos. Fast nearest-neighbor query processing in moving-object databases. *GeoInformatica*, 7(2):113–137, 2003.

[75] Glenn Simmons Iwerks, Hanan Samet, and Kenneth P. Smith. Continuous k-nearest neighbor queries for continuously moving points with updates. In *VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany*, pages 512–523, 2003.

[76] Yufei Tao, Christos Faloutsos, Dimitris Papadias, and Bin Liu. Prediction and indexing of moving objects with unknown motion patterns. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 611–622, 2004.

[77] Norio Katayama and Shin'ichi Satoh. The sr-tree: An index structure for high-dimensional nearest neighbor queries. *ACM Sigmod Record*, 26(2):369–380, 1997.

[78] Wei Wang, Jiong Yang, and Richard Muntz. Pk-tree: a spatial index structure for high dimensional point data. In *Information organization and databases*, pages 281–293. Springer, 2000.

[79] Xiaolin Wang, Yingwei Luo, Lishan Yu, and Zhuoqun Xu. Pk+ tree: an improved spatial index structure of pk tree. In *International Conference on Computational Science*, pages 511–514. Springer, 2005.

[80] Tenindra Abeywickrama, Muhammad Aamir Cheema, and David Taniar. K-nearest neighbors on road networks: a journey in experimentation and in-memory implementation. *arXiv preprint arXiv:1601.01549*, 2016.

[81] Andrew V Goldberg and Chris Harrelson. Computing the shortest path: A search meets graph theory. In *SODA*, volume 5, pages 156–165, 2005.

[82] Hector Gonzalez, Jiawei Han, Xiaolei Li, Margaret Myslinska, and John Paul Sondag. Adaptive fastest path computation on a road network: a traffic mining approach. In *33rd International Conference on Very Large Data Bases, VLDB 2007*, pages 794–805. Association for Computing Machinery, Inc, 2007.

[83] Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[84] Daniel R Lanning, Gregory K Harrell, and Jin Wang. Dijkstra's algorithm and google maps. In *Proceedings of the 2014 ACM Southeast Regional Conference*, pages 1–3, 2014.

[85] David Taniar, Maytham Safar, Quoc Thai Tran, Wenny Rahayu, and Jong Hyuk Park. Spatial network rnn queries in gis. *The Computer Journal*, 54(4):617–627, 2011.

[86] Shuo Shang, Bo Yuan, Ke Deng, Kexin Xie, and Xiaofang Zhou. Finding the most accessible locations: reverse path nearest neighbor query in road networks. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 181–190, 2011.

[87] Xiao-Jun Xu, Jin-Song Bao, Bin Yao, Jing-Yu Zhou, Fei-Long Tang, Min-Yi Guo, and Jian-Qiu Xu. Reverse furthest neighbors query in road networks. *Journal of Computer Science and Technology*, 32(1):155–167, 2017.

[88] Muhammad Aamir Cheema, Xuemin Lin, Wei Wang, Wenjie Zhang, and Jian Pei. Probabilistic reverse nearest neighbor queries on uncertain data. *IEEE Trans. Knowl. Data Eng.*, 22(4):550–564, 2010.

[89] Tobias Emrich, Hans-Peter Kriegel, Peer Kröger, Matthias Renz, and Andreas Züfle. Incremental reverse nearest neighbor ranking in vector spaces. In *SSTD*, pages 265–282. Springer, 2009.

[90] Amit Singh, Hakan Ferhatosmanoglu, and Ali Şaman Tosun. High dimensional reverse nearest neighbor queries. In *CIKM*, pages 91–98. ACM, 2003.

[91] Ioana Stanoi, Divyakant Agrawal, and Amr El Abbadi. Reverse nearest neighbor queries for dynamic databases. In *ACM SIGMOD workshop on research issues in data mining and knowledge discovery*, pages 44–53, 2000.

[92] Wei Wu, Fei Yang, Chee-Yong Chan, and Kian-Lee Tan. Finch: Evaluating reverse k-nearest-neighbor queries on location data. *PVLDB*, 1(1):1056–1067, 2008.

[93] Yufei Tao, Dimitris Papadias, and Xiang Lian. Reverse knn search in arbitrary dimensionality. In *VLDB*, volume 30, pages 744–755. VLDB Endowment, 2004.

[94] Elke Achtert, Hans-Peter Kriegel, Peer Kröger, Matthias Renz, and Andreas Züfle. Reverse k-nearest neighbor search in dynamic and general metric databases. In *EDBT*, pages 886–897. ACM, 2009.

[95] Ioana Stanoi, Mirek Riedewald, Divyakant Agrawal, and Amr El Abbadi. Discovery of influence sets in frequently updated databases. In *VLDB*, volume 2001, pages 99–108, 2001.

[96] Muhammad Aamir Cheema, Wenjie Zhang, Xuemin Lin, Ying Zhang, and Xuefei Li. Continuous reverse k nearest neighbors queries in euclidean space and in spatial networks. *VLDB J.*, 21(1):69–95, 2012.

[97] Muhammad Aamir Cheema, Xuemin Lin, Ying Zhang, Wei Wang, and Wenjie Zhang. Lazy updates: An efficient technique to continuously monitoring reverse knn. *PVLDB*, 2(1):1138–1149, 2009.

[98] Thomas Bernecker, Tobias Emrich, Hans-Peter Kriegel, Matthias Renz, Stefan Zankl, and Andreas Züfle. Efficient probabilistic reverse nearest neighbor query processing on uncertain data. *PVLDB*, 4(10):669–680, 2011.

[99] Shiyu Yang, Muhammad Aamir Cheema, Xuemin Lin, and Ying Zhang. Slice: Reviving regions-based pruning for reverse k nearest neighbors queries. In *ICDE*, pages 760–771. IEEE, 2014.

[100] Flip Korn and Suresh Muthukrishnan. Influence sets based on reverse nearest neighbor queries. *ACM Sigmod Record*, 29(2):201–212, 2000.

[101] Flip Korn and S. Muthukrishnan. Influence sets based on reverse nearest neighbor queries. In *SIGMOD*, pages 201–212, 2000.

[102] Bohan Li and Xiaolin Qin. Research on reverse nearest neighbor queries using ranked voronoi diagram. In *2009 First International Conference on Information Science and Engineering*, pages 951–955. IEEE, 2009.

[103] Geng Zhao, Kefeng Xuan, Wenny Rahayu, David Taniar, Maytham Safar, Marina L Gavrilova, and Bala Srinivasan. Voronoi-based continuous k nearest neighbor search in mobile navigation. *IEEE Transactions on Industrial Electronics*, 58(6):2247–2257, 2009.

[104] Shiyu Yang, Muhammad Aamir Cheema, Xuemin Lin, and Wei Wang. Reverse k nearest neighbors query processing: experiments and analysis. *PVLDB*, 8(5):605–616, 2015.

[105] Ioana Stanoi, Divyakant Agrawal, and Amr El Abbadi. Reverse nearest neighbor queries for dynamic databases. In *ACM SIGMOD Workshop*, 2000.

[106] Yufei Tao, Dimitris Papadias, and Xiang Lian. Reverse knn search in arbitrary dimensionality. *PVLDB*, pages 744–755, 2004.

[107] Shiyu Yang, Muhammad Aamir Cheema, Xuemin Lin, and Ying Zhang. SLICE: Reviving regions-based pruning for reverse k nearest neighbors queries. In *ICDE*, pages 760–771, 2014.

[108] Wei Wu, Fei Yang, Chee Yong Chan, and Kian-Lee Tan. Continuous reverse k-nearest-neighbor monitoring. In *MDM*, pages 132–139, 2008.

[109] AL-Khalidi Haidar, David Taniar, and Maytham Safar. Approximate algorithms for static and continuous range queries in mobile navigation. *Computing*, 95(10-11):949–976, 2013.

[110] Mingjin Zhang, Naphtali Rishe, LIU Weitong, Jahkell Lazarre, and Tao Li. Voronoi diagram-based algorithm for efficient progressive continuous k-nearest neighbor query for moving objects, February 5 2019. US Patent App. 10/200,814.

[111] Rimantas Benetis, Christian S Jensen, Gytis Karciauskas, and Simonas Saltenis. Nearest neighbor and reverse nearest neighbor queries for moving objects. In *Database Engineering and Applications Symposium*, pages 44–53. IEEE, 2002.

[112] Tian Xia and Donghui Zhang. Continuous reverse nearest neighbor monitoring. In *ICDE*, pages 77–77. IEEE, 2006.

[113] James M Kang, Mohamed F Mokbel, Shashi Shekhar, Tian Xia, and Donghui Zhang. Continuous evaluation of monochromatic and bichromatic reverse nearest neighbors. In *ICDE*, pages 806–815. IEEE, 2007.

[114] Tobias Emrich, Hans-Peter Kriegel, Peer Kröger, Matthias Renz, Naixin Xu, and Andreas Züfle. Reverse k-nearest neighbor monitoring on mobile objects. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 494–497, 2010.

[115] Man Lung Yiu, Dimitris Papadias, Nikos Mamoulis, and Yufei Tao. Reverse nearest neighbors in large graphs. *IEEE Transactions on Knowledge and Data Engineering*, 18(4):540–553, 2006.

[116] Elke Achtert, Christian Böhm, Peer Kröger, Peter Kunath, Alexey Pryakhin, and Matthias Renz. Efficient reverse k-nearest neighbor search in arbitrary metric spaces. In *SIGMOD Conference*, pages 515–526, 2006.

[117] Kefeng Xuan, Geng Zhao, David Taniar, Bala Srinivasan, Maytham Safar, and Marina Gavrilova. Network voronoi diagram based range search. In *2009 International Conference on Advanced Information Networking and Applications*, pages 741–748. IEEE, 2009.

[118] Bolong Zheng, Kai Zheng, Xiaokui Xiao, Han Su, Hongzhi Yin, Xiaofang Zhou, and Guohui Li. Keyword-aware continuous knn query on road networks. In *2016 IEEE 32Nd international conference on data engineering (ICDE)*, pages 871–882. IEEE, 2016.

[119] Man Lung Yiu, Dimitris Papadias, Nikos Mamoulis, and Yufei Tao. Reverse nearest neighbors in large graphs. In *ICDE*, 2005.

[120] Yufei Tao, Dimitris Papadias, and Qiongmao Shen. Continuous nearest neighbor search. In *VLDB*, pages 287–298, 2002.

[121] 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, San Diego, California, USA, June 9-12, 2003*, pages 443–454, 2003.

[122] Mohamed F. Mokbel, Xiaopeng Xiong, and Walid G. Aref. Sina: Scalable incremental processing of continuous queries in spatio-temporal databases. In *SIGMOD Conference*, pages 623–634, 2004.

[123] Xiaopeng Xiong, Mohamed F. Mokbel, and Walid G. Aref. Sea-cnn: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In *ICDE*, pages 643–654, 2005.

[124] Kyriakos Mouratidis, Marios Hadjieleftheriou, and Dimitris Papadias. Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, pages 634–645, 2005.

[125] Xiaohui Yu, Ken Q. Pu, and Nick Koudas. Monitoring k-nearest neighbor queries over moving objects. In *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, 5-8 April 2005, Tokyo, Japan*, pages 631–642, 2005.

[126] Muhammad Aamir Cheema, Yidong Yuan, and Xuemin Lin. Circulartrip: An effective algorithm for continuous nn queries. In *DASFAA*, pages 863–869, 2007.

[127] Iosif Lazaridis, Kriengkrai Porkaew, and Sharad Mehrotra. Dynamic queries over mobile objects. In *EDBT*, pages 269–286, 2002.

[128] Ying Cai, Kien A. Hua, and Guohong Cao. Processing range-monitoring queries on heterogeneous mobile objects. In *Mobile Data Management*, 2004.

[129] Bugra Gedik and Ling Liu. Mobieyes: Distributed processing of continuously moving queries on moving objects in a mobile system. In *EDBT*, pages 67–87, 2004.

[130] Xiaoyuan Wang and Wei Wang. Continuous expansion: Efficient processing of continuous range monitoring in mobile environments. In *DASFAA*, pages 890–899, 2006.

[131] Haojun Wang, Roger Zimmermann, and Wei-Shinn Ku. Distributed continuous range query processing on moving objects. In *DEXA*, pages 655–665, 2006.

[132] Tian Xia and Donghui Zhang. Continuous reverse nearest neighbor monitoring. In *ICDE*, pages 77–86, 2006.

[133] James M. Kang, Mohamed F. Mokbel, Shashi Shekhar, Tian Xia, and Donghui Zhang. Continuous evaluation of monochromatic and bichromatic reverse nearest neighbors. In *ICDE*, pages 806–815, 2007.

[134] Muhammad Aamir Cheema, Wenjie Zhang, Xuemin Lin, Ying Zhang, and Xuefei Li. Continuous reverse k nearest neighbors queries in euclidean space and in spatial networks. *VLDB J.*, pages 69–95, 2012.

[135] Haibo Hu, Jianliang Xu, and Dik Lun Lee. A generic framework for monitoring continuous spatial queries over moving objects. In *SIGMOD Conference*, pages 479–490, 2005.

[136] Kyriakos Mouratidis, Dimitris Papadias, Spiridon Bakiras, and Yufei Tao. A threshold-based algorithm for continuous monitoring of k nearest neighbors. *TKDE*, pages 1451–1464, 2005.

[137] Muhammad Aamir Cheema, Xuemin Lin, Ying Zhang, Wei Wang, and Wenjie Zhang. Lazy updates: An efficient technique to continuously monitoring reverse knn. *PVLDB*, pages 1138–1149, 2009.

[138] Tobias Emrich, Hans-Peter Kriegel, Peer Kröger, Matthias Renz, Naixin Xu, and Andreas Züfle. Reverse k-nearest neighbor monitoring on mobile objects. In *GIS*, pages 494–497, 2010.

[139] Sujin Oh, HaRim Jung, and Ung-Mo Kim. An efficient processing of range spatial keyword queries over moving objects. In *International Conference on Information Networking*, pages 525–530. IEEE, 2018.

[140] Saad Aljubayrin, Jianzhong Qi, Christian S Jensen, Rui Zhang, Zhen He, and Zeyi Wen. The safest path via safe zones. In *ICDE*, pages 531–542. IEEE, 2015.

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

[142] Jun Zhang, Manli Zhu, Dimitris Papadias, Yufei Tao, and Dik Lun Lee. Location-based spatial queries. In *SIGMOD*, pages 443–454. ACM, 2003.

[143] Muhammad Aamir Cheema, Ljiljana Brankovic, Xuemin Lin, Wenjie Zhang, and Wei Wang. Continuous monitoring of distance-based range queries. *IEEE Trans. Knowl. Data Eng.*, 23(8):1182–1199, 2011.

[144] AL-Khalidi Haidar, David Taniar, John Betts, and Sultan Alamri. On finding safe regions for moving range queries. *Mathematical and Computer Modelling*, 58(5-6):1449–1458, 2013.

[145] Haibo Hu, Jianliang Xu, and Dik Lun Lee. A generic framework for monitoring continuous spatial queries over moving objects. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 479–490, 2005.

[146] Hyung-Ju Cho, Se Jin Kwon, and Tae-Sun Chung. A safe exit algorithm for continuous nearest neighbor monitoring in road networks. *Mobile Information Systems*, 9(1):37–53, 2013.

[147] Ch N Santhosh Kumar, V Sitha Ramulu, K Sudheer Reddy, Suresh Kotha, and Ch Mohan Kumar. Spatial data mining using cluster analysis. *International Journal of Computer Science & Information Technology*, 4(4):71, 2012.

[148] M Emre Celebi, Hassan A Kingravi, and Patricio A Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200–210, 2013.

[149] Raymond T. Ng and Jiawei Han. Clarans: A method for clustering objects for spatial data mining. *IEEE Trans. Knowl. Data Eng.*, 14(5):1003–1016, 2002.

[150] NS Sneha et al. Clustering and noise detection for geographic knowledge discovery. *International Journal of Advances in Engineering & Technology*, 7(3):845, 2014.

[151] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017.

[152] Thao P Nghiem, Kiki Maulana, Agustinus Borgy Waluyo, David Green, and David Taniar. Bichromatic reverse nearest neighbors in mobile peer-to-peer

networks. In *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 160–165. IEEE, 2013.

[153] Thao P Nghiem, David Green, and David Taniar. Peer-to-peer group k-nearest neighbours in mobile ad-hoc networks. In *2013 International Conference on Parallel and Distributed Systems*, pages 166–173. IEEE, 2013.

[154] Dong-Wan Choi and Chin-Wan Chung. Nearest neighborhood search in spatial databases. In *ICDE*, pages 699–710. IEEE, 2015.

[155] Feifei Li, Bin Yao, and Piyush Kumar. Group enclosing queries. *IEEE Transactions on Knowledge and Data Engineering*, 23(10):1526–1540, 2010.

[156] Hongga Li, Hua Lu, Bo Huang, and Zhiyong Huang. Two ellipse-based pruning methods for group nearest neighbor queries. In *Proceedings of the 13th annual ACM international workshop on Geographic information systems*, pages 192–199, 2005.

[157] Ke Deng, Shazia Sadiq, Xiaofang Zhou, Hu Xu, Gabriel Pui Cheong Fung, and Yansheng Lu. On group nearest group query processing. *IEEE Transactions on Knowledge and Data Engineering*, 24(2):295–308, 2010.

[158] Alok Aggarwal, Hiroshi Imai, Naoki Katoh, and Subhash Suri. Finding k points with minimum diameter and related problems. *Journal of algorithms*, 12(1):38–56, 1991.

[159] Yongshan Liu, Xiang Gong, Dehan Kong, Tianbao Hao, and Xiaoqi Yan. A voronoi-based group reverse k farthest neighbor query method in the obstacle space. *IEEE Access*, 8:50659–50673, 2020.

[160] Hiba Jadallah and Zaher Al Aghbari. Spatial cloaking for location-based queries in the cloud. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–9, 2018.

[161] Quoc Thai Tran, David Taniar, and Maytham Safar. Bichromatic reverse nearest-neighbor search in mobile systems. *IEEE Systems Journal*, 4(2):230–242, 2010.

[162] David Taniar and Wenny Rahayu. A taxonomy for nearest neighbour queries in spatial databases. *J. Comput. Syst. Sci.*, 79(7):1017–1039, 2013.

[163] Robert Lübke, Daniel Schuster, and Alexander Schill. Mobilisgroups: Location-based group formation in mobile social networks. In *PERCOM Workshops*, pages 502–507. IEEE, 2011.

[164] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: an efficient and robust access method for points and rectangles. In *Acm Sigmod Record*, volume 19, pages 322–331. Acm, 1990.

[165] Abdelkader Hameurlain, Josef Küng, and Roland Wagner. *Transactions on Large-Scale Data-and Knowledge-Centered Systems I*, volume 5740. Springer, 2009.

[166] Huang-Chen Lee and Kai-Hsiang Ke. Monitoring of large-area iot sensors using a lora wireless mesh network system: Design and evaluation. *IEEE Trans. Instrumentation and Measurement*, 67(9):2177–2187, 2018.

[167] Brian D Ripley. *Stochastic simulation*, volume 316. John Wiley & Sons, 2009.

[168] Thomas Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.

[169] Thao P Nghiem, Agustinus Borgy Waluyo, and David Taniar. A pure peer-to-peer approach for knn query processing in mobile ad hoc networks. *Personal and Ubiquitous Computing*, 17(5):973–985, 2013.

[170] Ken CK Lee, Wang-Chien Lee, Baihua Zheng, and Yuan Tian. Road: A new spatial object search framework for road networks. *IEEE transactions on knowledge and data engineering*, 24(3):547–560, 2010.

[171] Ken CK Lee, Wang-Chien Lee, and Baihua Zheng. Fast object search on road networks. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 1018–1029, 2009.

[172] Weimo Liu, Weiwei Sun, Chunan Chen, Yan Huang, Yinan Jing, and Kunjie Chen. Circle of friend query in geo-social networks. In *International Conference on Database Systems for Advanced Applications*, pages 126–137. Springer, 2012.

[173] Akrivi Vlachou, Christos Doulkeridis, Yannis Kotidis, and Kjetil Nørvåg. Reverse top-k queries. In *ICDE*, pages 365–376, 2010.

[174] Muhammad Aamir Cheema, Zhitao Shen, Xuemin Lin, and Wenjie Zhang. A unified framework for efficiently processing ranking related queries. In *EDBT*, 2014.

[175] Shiyu Yang, Muhammad Aamir Cheema, Xuemin Lin, Ying Zhang, and Wenjie Zhang. Reverse k nearest neighbors queries and spatial reverse top-k queries. In *VLDB Journal*, 2016.

Appendix A

During our experimental investigation into Reverse Nearest Neighbourhood queries on Euclidean distance (detailed in Chapter 3 and 4), we used two types of datasets, synthetic and real dataset. For synthetic dataset, we used C++ code to generate synthetic datasets with variety of data sizes and normal distribution setting, each data set is indexed by an R*-tree with the page size of 4. The generated dataset has three columns- ID, longitude and latitude for each point of interest. The data is normalized between 0-1 as shown in Figure A.1.

1	0.748125970363617	0.229037821292877
2	0.338308483362198	0.335045546293259
3	0.721539914608002	0.317346900701523
4	0.222703918814659	0.490478724241257
5	0.550526857376099	0.724517405033112
6	0.625650942325592	0.253315269947052
7	0.486361622810364	0.326119691133499
8	0.301749497652054	0.459169477224350
9	0.376199603080750	0.665347158908844
10	0.673875570297241	0.448640316724777
11	0.469123125076294	0.722101151943207
12	0.333755105733871	0.565716922283173
13	0.230212882161140	0.585389614105225
14	0.458537340164185	0.483713239431381
15	0.481364995241165	0.356504291296005
16	0.709463298320770	0.594557523727417
17	0.473881095647812	0.705147981643677
18	0.718767344951630	0.618343114852905
19	0.349302917718887	0.571881115436554
20	0.636761486530304	0.480842351913452
21	0.464066505432129	0.444099754095078
22	0.750203132629395	0.898483276367188
....	.....	.....
....	.....	.....
....	.....	.....

Figure A.1: Sample of synthetic dataset

For real dataset, we obtain the North America, Los Angeles and Californian dataset from http://www.cs.utah.edu/ lifeifei/SpatialDataset.htm. The North America dataset consists of 175, 812 points, the Los Angeles and California datasets contain 2.6 million points and around 25.8 million points Respectively. Also, we used the code to normalize the real dataset between 0-1 to run the experiment.

Appendix B

During our experimental investigation into Reverse Nearest Neighbourhood queries on Road Networks, (detailed in Chapter 5) we used real dataset in Melbourne city, Australia. In this appendix section, we describe the dataset and provide some sample datasets that we use in our experiments in the study of Reverse Nearest neighbourhood query on Road Networks.

We used OpenStreetMap (OSM) to get the geographical data (real dataset e.g Melbourne City). OpenStreetMap (OMS) provides global map data in a unified tagging schema. The Data of OSM is available for every country in the world, and the data quality is excellent. we have chosen OSM because it is free data and open source, it is possible for anybody to download all the data in this database, Figure shows the sample of the synthetic detest.

```
<?xml version="1.0" encoding="utf-8" ?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document id="root_doc">
<Schema name="Clayton" id="Clayton">
    <SimpleField name="SgtID" type="string"></SimpleField>
    <SimpleField name="route" type="string"></SimpleField>
</Schema>
<Folder><name>Clayton</name>
  <Placemark>
    <Style><LineStyle><color>ff0000ff</color></LineStyle><PolyStyle><fill>0</fill></PolyStyle></Style>
    <ExtendedData><SchemaData schemaUrl="#Clayton">
        <SimpleData name="SgtID">1A1A1A1A</SimpleData>
        <SimpleData name="route">Greenways Road</SimpleData>
    </SchemaData></ExtendedData>
      <LineString><coordinates>145.1491638,-37.8831865 145.1494803,-37.8829706 </coordinates></LineString>
  </Placemark>
  <Placemark>
    <Style><LineStyle><color>ff0000ff</color></LineStyle><PolyStyle><fill>0</fill></PolyStyle></Style>
    <ExtendedData><SchemaData schemaUrl="#Clayton">
        <SimpleData name="SgtID">1A1A1A1A</SimpleData>
        <SimpleData name="route">Greenways Road</SimpleData>
    </SchemaData></ExtendedData>
      <LineString><coordinates>145.1494803,-37.8829706 145.1496773,-37.8829519 </coordinates></LineString>
  </Placemark>
  <Placemark>
    <Style><LineStyle><color>ff0000ff</color></LineStyle><PolyStyle><fill>0</fill></PolyStyle></Style>
    <ExtendedData><SchemaData schemaUrl="#Clayton">
        <SimpleData name="SgtID">1A1A1A1A</SimpleData>
        <SimpleData name="route">Greenways Road</SimpleData>
    </SchemaData></ExtendedData>
      <LineString><coordinates>145.1496773,-37.8829519 145.1505624,-37.883062 </coordinates></LineString>
  </Placemark>
  <Placemark>
    <Style><LineStyle><color>ff0000ff</color></LineStyle><PolyStyle><fill>0</fill></PolyStyle></Style>
    <ExtendedData><SchemaData schemaUrl="#Clayton">
        <SimpleData name="SgtID">1A1A1A1A</SimpleData>
        <SimpleData name="route">Greenways Road</SimpleData>
    </SchemaData></ExtendedData>
      <LineString><coordinates>145.1505624,-37.883062 145.1511042,-37.8831731 </coordinates></LineString>
  </Placemark>
```

Figure B.1: kml file

OSM is built by a community of mappers that contribute and maintain data about roads, trails, cafés, railway stations, and much more, all over the world. Because it is so massive (the data is more than 30 GB even when it's compressed), it's nearly impossible to work with all the data at once, we used every map and dataset separately and we did our experiment separately to report the outcomes. To get our customized up-to-date OSM data, we used QGIS (formerly Quantum GIS),

which is a full-featured, open-source, cross-platform Geographic Information System, with QGIS we have access up-to-date OSM data. OSM provides the data in kml format as shown Figure B.1, kml is a file format used to display geographic data, it is a tag-based structure with nested elements and attributes, usually the kml has thousands of tags and all tags are case-sensitive.

```
void Kml_nodes_points()
{
      std::ofstream handle;

// http://www.cplusplus.com/reference/ios/ios/exceptions/
// Throw an exception on failure to open the file or on a write error.
handle.exceptions(std::ofstream::failbit | std::ofstream::badbit);

// Open the KML file for writing:
//handle.open("/home/nasseria/projects/GTree-master/src/gtree/Dataset7/
result_kml_Folder/nodes.kml");
handle.open(FILE_KML_ALLnodes);

// Write to the KML file:
handle << "<?xml version='1.0' encoding='utf-8'?>\n";
handle << "<kml xmlns='http://www.opengis.net/kml/2.2'>\n";
handle << "<Document id='root_doc'> \n" ;
handle << " <Schema name= 'points' id='points'>\n <SimpleField name='users'
type='string'></SimpleField>\n <SimpleField name='facilites' type='string'></
SimpleField> </Schema> \n" ;
handle << "<Folder><name>Nodes_pointes</name>\n";

for(int i =0 ; i<Nodes.size() ; i++)
{
    Node node_1 =  returen_Node_INFO (Nodes[i].ID ) ;
   double lat1 = node_1.y  ;
   double  long1= node_1.x  ;
   int   id = node_1.ID ;
    handle << FormatPlacemark( lat1,  long1, id) ;

}
 handle << "</Folder>\n";
handle << "</Document></kml>\n";
handle.close();
}
```

Figure B.2: Snapshot code 1

In our project, we need to convert OpenStreetMap data from its native format (i.e kml format) into a format that is convenient for our code (i.e is csv format). The most important tag is 'Coordinates' tag, which has longitude and latitude for very vertex (also called node). This tag what we exactly need to use in our code to do the experiment. In order to represent graphs on our code, we need to have two files from OSM: Edges file and node file. The edges file contains the list of connected vertex points that represent the graph of road network. The node file is the file has the Coordinates' points of interest. The challenge here is how convert these data from kml format to csv format to allow the code to read these geographical data. We conducted several codes, to convert the data from kml format to csv format, and also convert result back to kml format to display it on QGIS, sample of codes attached below.

```
void Kml_users_points()
{
    cout<< "Welcome to Kml_users_points "<<endl;
      std::ofstream handle;
// http://www.cplusplus.com/reference/ios/ios/exceptions/
// Throw an exception on failure to open the file or on a write error.
handle.exceptions(std::ofstream::failbit | std::ofstream::badbit);

// Open the KML file for writing:
handle.open(FILE_KML_users);

// Write to the KML file:
handle << "<?xml version='1.0' encoding='utf-8'?>\n";
handle << "<kml xmlns='http://www.opengis.net/kml/2.2'>\n";
handle << "<Document id='root_doc'> \n" ;
handle << " <Schema name= 'points' id='points'>\n <SimpleField name='owner'
type='string'></SimpleField>\n <SimpleField name='oid' type='string'></
SimpleField>\n <SimpleField name='RNN' type='string'></SimpleField>\n </
Schema> \n   " ;
handle << "<Folder><name>users-WithRNN</name>\n";

int oid = 1 ;
for(int i=0; i < user_node.size(); i++)
    {
        Node node_1 =  returen_Node_INFO (user_node[i] ) ;

                double lat1 = node_1.y  ;
                double  long1= node_1.x  ;
                int  nid = node_1.ID ;
                int  index_node =  indexNodes.find(nid)->second;
                int NF =  Nodes[index_node].NF ;

                handle << FormatPlacemark_with_object_id(lat1,
                long1,nid,oid,NF);

                oid++ ;

    }
handle << "</Folder>\n";
handle << "</Document></kml>\n";
handle.close();
}
```

Figure B.3: Snapshot code 2

```
void Kml_faciltes_points()
{
     cout<< "Welcome to Kml_faciltes_points  "<<endl;
     std::ofstream handle;
// http://www.cplusplus.com/reference/ios/ios/exceptions/
// Throw an exception on failure to open the file or on a write error.
handle.exceptions(std::ofstream::failbit | std::ofstream::badbit);

// Open the KML file for writing:
handle.open(FILE_KML_facilites);

// Write to the KML file:
handle << "<?xml version='1.0' encoding='utf-8'?>\n";
handle << "<kml xmlns='http://www.opengis.net/kml/2.2'>\n";
handle << "<Document id='root_doc'> \n" ;
handle << " <Schema name= 'points' id='points'>\n <SimpleField name='owner'
type='string'></SimpleField>\n <SimpleField name='oid' type='string'></
SimpleField> </Schema> \n" ;

handle << "<Folder><name>facility</name>\n";
int oid= 1 ;
for(int i=0; i < Facility_node.size(); i++)
    {
        Node node_1 =  returen_Node_INFO (Facility_node[i] ) ;

                double lat1 = node_1.y  ;
                double  long1= node_1.x  ;
                int nid = node_1.ID ;
                //handle << FormatPlacemark(lat1, long1,nid);
                handle << FormatPlacemark_with_object_id_Facility(lat1,
                long1,nid,oid);
                oid++ ;

    }
handle << "</Folder>\n";
handle << "</Document></kml>\n";
handle.close();
}
```

Figure B.4: Snapshot code 3

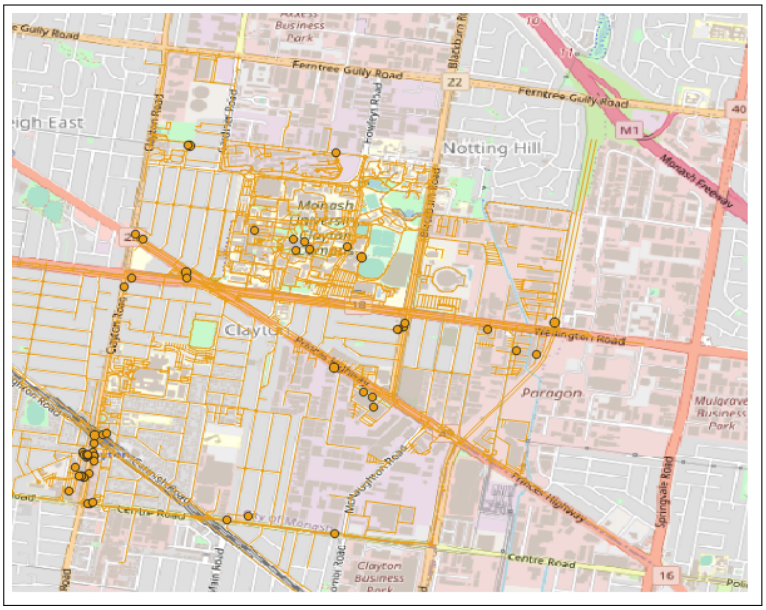


Figure B.5: Monash University Map

B.1 Monash University Dataset

In this section, we provide some sample of data that we use in our experiments for the area of Monash University and residential area. This dataset has approximately 6600 edges and 8055 vertices. It consists of vertices data, edges data and name of roads data. We calculate the bounding box. That is, the coordinate for the south-west and north-east corners for each dataset. The bounding box of this dataset is [[[145.1186014135, -37.9320371072],[145.1446971548,-37.9320371072],[145.1446971548,-37.9052954916], [145.1186014135,-37.9052954916],[145.1186014135,-37.9320371072]]].

B.1.1 Sample data for vertices

Below we present sample vertices data. The first line indicates the number of vertices in the dataset, the second line and forward consists of ID vertex, the longitude and latitude of the corresponding vertex.

Table B.1: Sample data for vertices

8055		
1	145.1259714	-37.9054102
2	145.1260067	-37.905411
3	145.1281139	-37.9056567
4	145.125709	-37.9063147
5	145.1279413	-37.9065627
7	145.125899	-37.9054271
8	145.1258534	-37.9055065
9	145.1257992	-37.9057933
11	145.1256043	-37.9068522
12	145.1231957	-37.9065771
13	145.1240192	-37.9066723
14	145.1243115	-37.9067062
15	145.1253951	-37.9068268
17	145.1264895	-37.9069517
18	145.1275972	-37.9070744
19	145.1278386	-37.9071019
20	145.1369405	-37.9060953
21	145.1356388	-37.9059476
22	145.1354394	-37.905925
23	145.1348515	-37.9058589
24	145.1346786	-37.9058395
.....	.....	.....
.....	.....	.....
.....	.....	.....
8055	145.135371	-37.9305305

B.1.2 Sample data for edges

Below we present sample edges data. The data shows the vertex where the edge start, the vertex where the edges ends and the edges weight.

Table B.2: Sample data for edges

startVert	**endVert**	**weight**
1	2	0.003099
1	7	0.006626
1	1985	0.005889
2	3	0.190015
3	2009	0.560648
3	2011	0.595481
4	5	0.197837
4	9	0.103206
4	11	0.163592
5	2011	0.662551
5	19	0.723197
7	8	0.014899
8	9	0.045211
9	1330	0.184284
11	15	0.213579
11	17	0.292048
12	13	0.073041
12	5053	0.712266
12	5055	0.840024
13	14	0.098969
13	1315	0.024374
14	15	0.195005
14	5085	0.267418
15	5091	0.267552
17	18	0.390211
17	5093	0.267219
18	19	0.411615
18	5097	0.267199
19	5029	0.077065
...	....	.....
...	....	.....

B.1.3 Sample data for users

Below we present sample user data. The data shows the longitude and latitude for users' dataset.

Table B.3: Sample data for users

LONGITUDE	LATITUDE
145.1221929	-37.91403247
145.1284275	-37.92967883
145.1193591	-37.92547642
145.1452244	-37.91733897
145.1471006	-37.91872568
145.1337721	-37.91211046
145.1185602	-37.9262554
145.125774	-37.9139819
145.1297546	-37.92942776
145.1301627	-37.91093746
145.1354394	-37.905925
145.1327241	-37.91152858
145.1483864	-37.91898264
145.1328783	-37.91222305
.....	.....
.....	.....
.....	.....
145.135371	-37.9305305

B.1.4 Sample data for facilities

Below we present sample facilities dataset. The data shows the longitude and latitude for facilities dataset.

Table B.4: Sample data for facilities

LONGITUDE	LATITUDE
145.1370723	-37.91266194
145.1353435	37.91979063
145.1257806	-37.9136487
145.1198387	-37.9242081
145.1495853	-37.91686526
145.1259714	-37.9054102

B.1.5 Data Analysis

In this section, we conducted geospatial analysis to describe and understand the behaviour of the dataset. We calculate the network distance between each point of interest (POI) to the furthest neighbour, and the network distance between each (POI) to the nearest neighbour.

Table B.5: Analysis for Furthest point Monash University

Distance from	**Number of POI**
1-5 km	1
5-6 km	8
6-7 km	16
7-8 km	7
8-9 km	28
The maximum distance: 8.90 km **Average distance to nearest shop: 7.44 km**	

Table B.6: Analysis for Nearest Neighbour Monash University

Distance from	**Number of POI**
1-100 m	34
200-300 m	2
300-400 m	7
400-600 m	7
600-700 m	2
700 m - 800 m	2
800 m - 900 m	2
above 1 km	4
The maximum distance : 2.11 km **Average distance to nearest shop: 0.3011 km**	

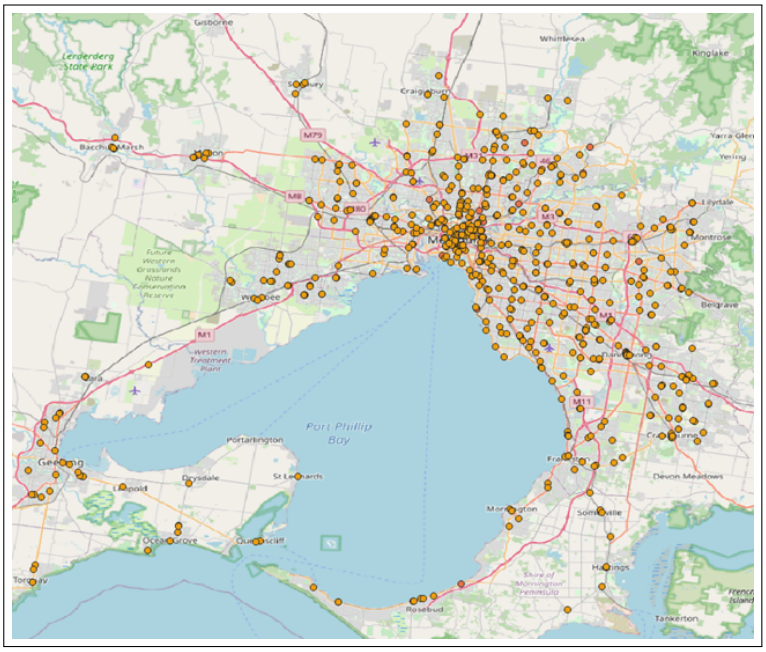


Figure B.6: Melbourne City Map

B.2 Melbourne City Dataset

In this section, we provide some sample of data that we use in our experiments for the Melbourne city dataset. This dataset has approximately contains more than 315500 vertices and 321107 edges. It consists of vertices data, edges data and name of roads data. The bounding box of Melbourne city's dataset is [[[144.2878134523, -38.4206323669],[145.2683432374,-38.4206323669],[145.2683432374,-37.6048346384],[144.2878134523, -37.6048346384],[144.2878134523,-38.4206323669]]].

B.2.1 Sample data for vertices

Below we present sample vertices data. The first line indicates the number of vertices in this dataset, the second line and forward consists of ID vertex, the longitude and latitude of the corresponding vertex.

Table B.7: Sample data for vertices

315500		
0	144.2504144	-37.0034686
1	144.2503271	-37.00347135
2	144.2502561	-37.00345926
3	144.2501989	-37.0034455
4	144.2501655	-37.00342298
5	146.3379059	-36.35170838
6	146.3383409	-36.35181624
7	146.342193	-36.35314537
8	146.3422708	-36.353174
9	146.345569	-36.35449675
10	146.3459701	-36.35465763
11	146.3463988	-36.35482951
12	146.3500773	-36.35648262
13	146.3531154	-36.35784776
14	146.3591908	-36.3605682
15	146.363331	-36.36236001
16	146.3637699	-36.36255487
17	146.364339	-36.36283619
18	146.3675099	-36.36440271
19	146.3696924	-36.36547879
20	146.3719234	-36.36658956
21	146.3745893	-36.36788101
22	146.3750524	-36.36810479
23	146.3753177	-36.36823549
24	146.3756826	-36.36841407
.....	.....	.....
.....	.....	.....
.....	.....	.....
315499	141.6148451	-38.20887904

B.2.2 Sample data for edges

In this section, we present a sample edges data. The data shows the vertex where the edge start, the vertex where the edges ends and the edges weight.

Table B.8: Sample data for edges

startVert	**endVert**	**weight**
0	19713	0.011482
0	207130	0.005365
1	2	0.00645
2	3	0.005308
3	4	0.003882
4	177827	0.006691
4	207100	0.016611
5	6	0.040774
5	255281	0.018637
6	7	0.375409
7	8	0.007661
8	9	0.330053
9	10	0.040143
10	11	0.042891
11	12	0.377341
12	13	0.311625
13	14	0.622661
14	15	0.420995
15	16	0.044886
16	17	0.059811
17	18	0.333202
18	19	0.229208
19	20	0.234923
20	21	0.278649
20	52431	0.012549
20	268887	0.048949
21	22	0.048369
22	23	0.027856
22	52416	0.402792
...	....	.....
...	....	.....

B.2.3 Sample data for users

Below we present a sample users data. The data shows the longitude and latitude for users' dataset in Melbourne City.

Table B.9: Sample data for users

LONGITUDE	LATITUDE
144.9764671	-37.78905832
145.1818192	-37.85544243
145.2411006	-37.87074421
145.2397342	-37.87029505
145.1644634	-37.83390631
145.1646846	-37.83393331
144.8915261	-38.35941223
144.8934445	-38.35887935
145.2794246	-37.79535168
145.2346475	-37.92029071
145.0816855	-37.88876456
145.0829908	-37.88969194
145.0817588	-37.88882966
145.2969608	-37.88828251
144.679175	-37.87708331
144.9669565	-37.7931284
145.249949	-37.89496698
144.9651327	-37.84972237
144.9395153	-37.81094778
144.5379905	-38.24558434
144.7140587	-37.59402499
145.0059933	-37.88907059
.....	.....
.....	.....
.....	.....
145.2945081	-38.04542953

B.2.4 Sample data for facilities

Below we present a sample facilities dataset. The data shows the longitude and latitude for facilities dataset.

Table B.10: Sample data for facilities

LONGITUDE	LATITUDE
145.2328663	-37.84688488
144.9878493	-37.91855824
145.1490436	-37.71270798
144.9624603	-37.77733085
144.9643987	-38.33291282
145.2506262	-38.0788901
145.0598959	-37.66723655
144.9912361	-37.77017074
145.0514896	-37.76028328
145.1595414	-37.67429572
145.238979	-38.45057132
144.9981074	-37.78633846
144.8276592	-37.78336412
145.0944776	-37.85028204
144.9975846	-37.81841658
.....	.....
.....	.....
.....	.....
144.9361564	-37.83727669

B.2.5 Data Analysis

In this section, we conducted geospatial analysis to describe and understand the behaviour of the Melbourne's dataset. We calculate the network distance between each point of interest (POI) to the furthest neighbour, and the network distance between each (POI) to the nearest neighbour.

Table B.11: Analysis for Furthest point Melbourne City

Distance up to (km)	**Number of POI**
60 km	51
70 km	210
80 km	136
90 km	110
100 km	71
more than 100 km	93
The maximum distance : 170 km **Average to furthest shop: 79 km**	

Table B.12: Analysis Nearest Neighbour Melbourne City

Distance up to (km)	**Number of POI**
100 M	350
200 M	29
300 M	38
500 M	23
800 M	40
1 km	14
more than 1 km	206
The maximum distance : 70 km **Average to nearest shop: 0.750 km**	

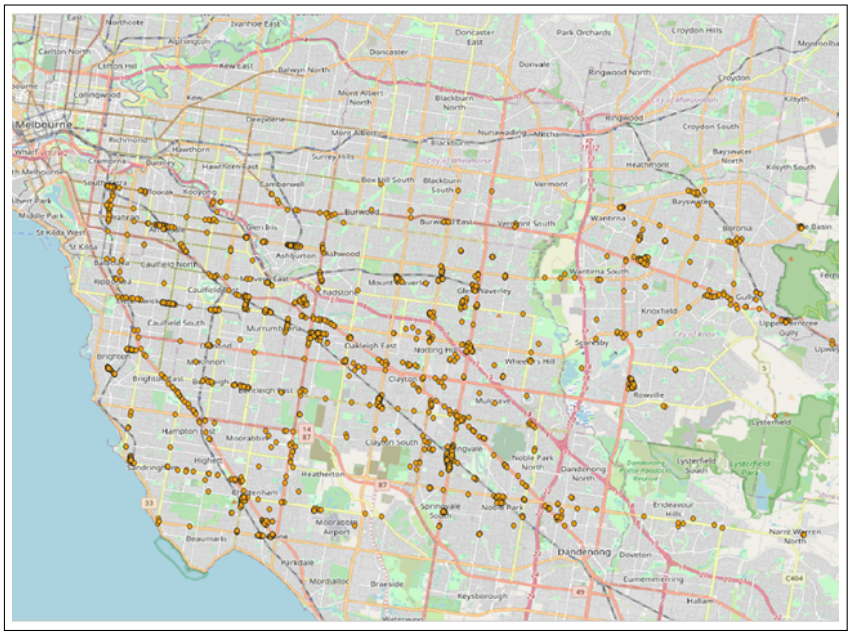


Figure B.7: South-East Melbourne Map

B.3 South-East Melbourne City Dataset

In this section, we provide some sample of data that we use in our experiments for the South-East Melbourne city. This dataset has approximately contains more than 195160 vertices and 215400 edges. It consists of vertices data, edges data and name of roads data. The bounding box of this dataset is [[[144.9881918703, -38.2180741131],[145.5649740968,-38.2180741131],[145.5649740968,-37.7504801667],[144.9881918703, -37.7504801667],[144.9881918703,-38.2180741131]]].

B.3.1 Sample data for vertices

Below we present sample vertices data. The first line indicates the number of vertices in the dataset, the second line and forward consists of vertex, an ID vertex, the longitude and latitude of the corresponding vertex.

Table B.13: Sample data for vertices

195160		
1	145.1491638	-37.8831865
2	145.1494803	-37.8829706
3	145.1496773	-37.8829519
4	145.1505624	-37.883062
5	145.1511042	-37.8831731
6	145.151752	-37.8832398
7	145.1523745	-37.8833077
8	145.1530343	-37.8833532
9	145.1550095	-37.8835849
10	145.1550712	-37.8836198
13	145.1550615	-37.8838192
14	145.15498	-37.8842189
15	145.1549425	-37.884293
18	145.1546367	-37.8842867
19	145.1542706	-37.8842644
20	145.1538924	-37.8842041
21	145.1535397	-37.8841374
22	145.1531789	-37.8841237
23	145.1526156	-37.8840474
24	145.152444	-37.884057
.....	.....	.....
.....	.....	.....
.....	.....	.....
195160	145.1320687	-37.8852786

B.3.2 Sample data for edges

Below we present a sample edges data. It shows the vertex where the edge start, the vertex where the edges ends and the edges weight for South-East Melbourne city.

Table B.14: Sample data for edges

startVert	**endVert**	**weight**
1	2	0.036724
1	34	0.232985
1	107	0.269166
2	3	0.052086
3	4	0.123557
4	5	0.170348
5	6	0.227288
6	7	0.28218
7	8	0.340285
7	75	0.335701
8	9	0.515085
9	10	0.520826
10	13	0.022195
13	14	0.067115
13	14527	0.006995
14	15	0.075725
15	18	0.026854
15	158696	0.020542
18	19	0.059069
19	20	0.092713
20	21	0.124356
21	22	0.15596
22	23	0.206087
23	24	0.220898
24	25	0.23537
25	26	0.241393
26	208442	0.0092
27	28	0.079224
27	63	0.135815
...	....	.....
...	....	.....

B.3.3 Sample data for users

In this section, we present a sample users data that shows the longitude and latitude for users' dataset in South-East Melbourne City.

Table B.15: Sample data for users

LONGITUDE	LATITUDE
145.1719519	-37.79025512
144.9764671	-37.78905832
145.1818192	-37.85544243
145.2411006	-37.87074421
145.2397342	-37.87029505
145.1644634	-37.83390631
145.1646846	-37.83393331
144.8915261	-38.35941223
144.8934445	-38.35887935
145.2794246	-37.79535168
145.2346475	-37.92029071
145.0816855	-37.88876456
145.0829908	-37.88969194
145.0817588	-37.88882966
145.2969608	-37.88828251
144.679175	-37.87708331
144.9669565	-37.7931284
145.249949	-37.89496698
145.2945081	-38.04542953
145.1488575	-37.85234353
145.3081223	-37.89355571
145.0022537	-37.95056345
145.1653185	-37.87421299
145.1662942	-37.87720266
145.3018675	-37.82230965
145.3101432	-37.80211338
145.310959	-37.80223016
.....	.....
.....	.....
.....	.....
144.5677605	-37.68573831

B.3.4 Sample Data for Facilities

Below we present a sample facilities dataset that shows the longitude and latitude for facilities dataset.

Table B.16: Sample data for facilities

LONGITUDE	LATITUDE
145.2328663	-37.84688488
144.9878493	-37.91855824
145.1490436	-37.71270798
144.9624603	-37.77733085
144.9643987	-38.33291282
145.2506262	-38.0788901
145.0598959	-37.66723655
144.9912361	-37.77017074
145.0514896	-37.76028328
145.1595414	-37.67429572
145.238979	-38.45057132
144.9981074	-37.78633846
144.8276592	-37.78336412
145.0944776	-37.85028204
144.9975846	-37.81841658
.....	.....
.....	.....
.....	.....
144.689448	-37.84210783
144.9361564	-37.83727669

B.3.5 Data Analysis

Also, we conducted geospatial analysis to describe and understand the behaviour of the Melbourne's dataset. We calculate the network distance between each point of interest (POI) to the furthest neighbour, and the network distance between each (POI) to the nearest neighbour.

Table B.17: Analysis for Furthest point South-East Melbourne

Distance up to (km)	**Number of POI**
less than 1 km	18
18 km	30
20 km	53
25 km	679
30 km	400
35 km	467
more than 35 km	16
The maximum distance : 37.23 km **Average distance distance to nearest shop: 25.6** km	

Table B.18: Analysis for Nearest Neighbour South-East Melbourne

Distance up to (km)	**Number of POI**
100 m	1458
200 m	108
300 m	61
400 m	29
700 m	31
1 km	13
6 km	36
The maximum distance : 5.13 km **Average distance to nearest shop: 0.1021 km**	

Last Thing

“Begin at the beginning and go on till you come to the end; then stop.“

“I can not go back to yesterday because I was a different person then.”

- Lewis Carroll