Received February 14, 2020, accepted February 27, 2020, date of publication March 9, 2020, date of current version March 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2979432

Safe Regions for Moving Reverse Neighbourhood Queries in a Peer-to-Peer Environment

NASSER ALLHEEIB[1], DAVID TANIAR[1], HAIDAR AL-KHALIDI[2], MD. SAIFUL ISLAM[3], AND KIKI MAULANA ADHINUGRAHA[4]

[1]Faculty of Information Technology, Monash University, Melbourne, VIC 3800, Australia
[2]Department of Software Systems and Cybersecurity, Monash University, Melbourne, VIC 3800, Australia
[3]School of Information and Communication Technology, Griffith University, Gold Coast Campus, Southport, QLD 4215, Australia
[4]Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3086, Australia

Corresponding author: Nasser Allheeib (nasser.allheeib@monash.edu)

ABSTRACT The peer-to-peer (P2P) paradigm has become very popular for storing and sharing information. 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 maximise the area known as the safe region (SR) where a long-range device can move freely while still maintaining connection with its peers. This paper illustrates an innovative and generic monitoring framework that addresses the issues related to frequent query location updating using a systematic approach. In our approach, we propose to 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. Monte-Carlo simulation was performed to calculate the area of the SR by weighing in shape irregularity. Experimental results demonstrate the effectiveness and efficiency of our approach.

INDEX TERMS Moving query, query processing, reverse nearest neighbourhood, safe region.

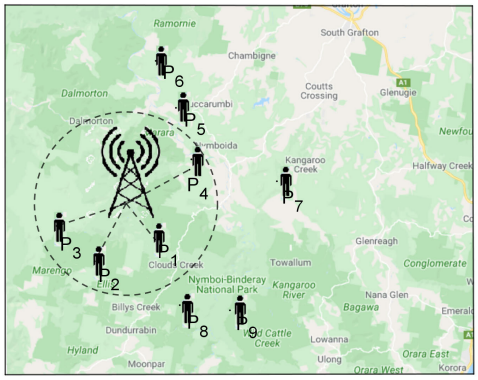
I. INTRODUCTION

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 allows them to broadcast their locations and share information via their mobile devices. Many of these systems are based on a centralised station [1], [2] (see Figure. 1); 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. 1. These centralised systems are susceptible to base station failures and isolated points issues.

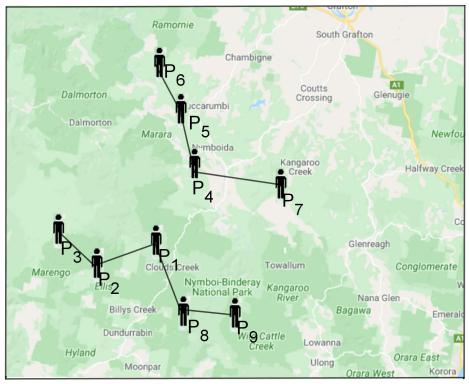
In response to the limitations of centralised systems, the emergence of mobile peer-to-peer (P2P) systems offers a promising solution. P2P overcomes the central failure (base station) and isolated points, where points can communicate or exchange information with their reachable surrounding points via short-range wireless communication (e.g. Bluetooth, Ad Hoc WiFi etc.) [3]. These interconnected peers in a P2P system form a neighbourhood [4]. Figure 1b 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 1b. 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

The associate editor coordinating the review of this manuscript and approving it for publication was waleed Alsabhan.



(a) Centralised systems



(b) P2P systems

FIGURE 1. **Centralised systems versus P2P systems.**

a LoRa wireless mesh network system by combining two types of communication: short-range and long-range [5]. With two types of communication, a system can overcome the problem of isolated neighbourhoods by enabling the two 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 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 [6]. During the Australian summer, dozens of fires burn across the country and residents brace themselves for catastrophic conditions [7]. In Figure 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 maximised. 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-rang 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.

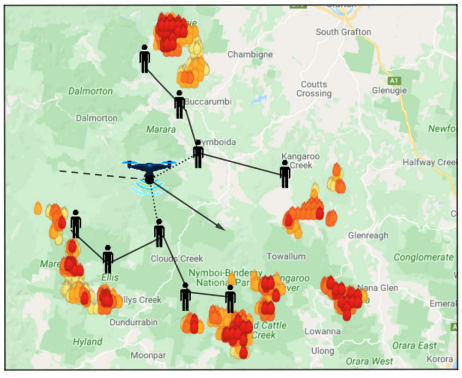


FIGURE 2. **Two types of communications in RNNH based P2P network systems.**

Contributions: The main contributions of this paper are summarised as follows.

- We introduce the concept of safe region (SR) for moving 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.

In our experiments, we apply 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.

The rest of the paper is organised as follows. The preliminaries and the related works are discussed in Sections II and III, respectively. In Section IV, the problem is defined formally and the SRs are formulated. Section V presents the calculation of area for the three types of SRs. Next, Section VI describes the empirical study, and Section VII concludes the paper.

II. PRELIMINARIES

A. DEFINITIONS

Consider a set of facilities $F = \{f_1, f_2, \ldots f_k\}$, a query point $q \in F$, a set of points $P = \{p_1, p_2, \ldots p_n\}$ as the points of interest. The facility points are the long-range network devices, and the points $p_i \in P$ are the short-range network devices. Query point q is the long-range network device that can move freely. In the figures, facilities are represented by triangles and points by (big) dots. We use the terms 'points' and 'devices' interchangeably in this paper and $dist(p_i, p_j)$ to denote the Euclidean distance between two distinct points p_i and p_j. Table 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 for 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 3, where $\{p_1, p_2, p_3, p_{10}, p_{11}, p_{17}\}$ are the only points that consider q as their nearest facility. The RNN query result is denoted by $RNN(q)$. On the other hand, the nearest neighbour (NN) 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 $NN(p_i)$.

Definition 1 (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 $NH(d, m)$.

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 $p_i \neq p_j$, $\forall i, j \in \{1, 2, \ldots, m\}$. 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 $NH(d, m)$.

TABLE 1. Notation.

Notation	Definition
q	the query point
$P = \{p_1, p_2, p_3, \ldots p_n\}$	a set of points, where n is a positive number
$F = \{f_1, f_2, f_3, \ldots f_k\}$	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 $NH(d, m)$
SOL	Safe Objects List, where $SOL \subset$P
F_d	the minimum Euclidean distance of the nearest competitor facility to a neighbourhood

Figure 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 $m \geq 3$.

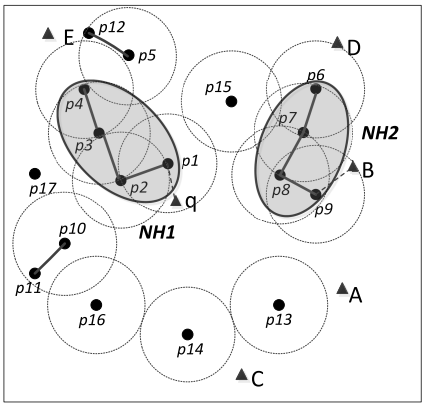


FIGURE 3. Example of neighbourhood queries $NH(d = 3, m = 3)$.

Definition 2 (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:

$$d_H(p_i, NH) = min\{dist(p_i, p_j) | p_j \in NH(d, m)\} \quad (1)$$

Figure 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 $dist(q, p_1)$.

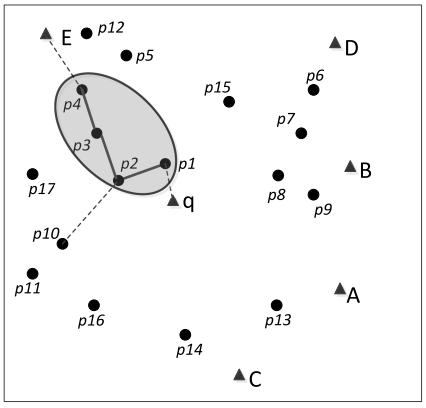


FIGURE 4. Neighbourhood distance.

Definition 3 (NNH Query): Given a a query facility q, a set of points P, two constraints d and m, and neighbourhoods $\mathcal{S} = \{NH_1, NH_2, \ldots, NH_{n'}\}$, where $NH_i \subseteq P$, $\forall i \in \{1, 2, \ldots n'\}$, 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. 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 4 (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 $\{NH_1, NH_2, \ldots, NH_{n'}\}$, 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, 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 $\{A, B, \ldots, E\}$ and the points $\{p_1, p_2, \ldots, p_{16}\}$ in the space as depicted in Figure 3. Here, the result of RNNH query for the query facility q is NH_1 for parameters $d = 3$ and $m = 3$ as $d_H(q, NH_1) \leq d_H(f, NH_1)$, $\forall f \in \{A, B, \ldots, E\}$. 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 nearest neighbourhood. Unlike an NNH query, with a RNNH query, we also need to deal with the competitor facilities.

B. RNNH QUERY PROCESSING

1) COMPLEXITY ANALYSIS

In a RNNH query, a neighbourhood NH must satisfy this: $d_H(q, NH) \leq d_H(f, NH)$, $\forall f \in F \setminus q$ to be a reverse nearest neighbourhood of q. Figure 4 displays an example of RNNH query result. A naïve RNNH 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 of the query q. In the example given in Figure 4, the distance between p_4 and its nearest facility E is $dist(p_4, E)$, which is greater than $d_H(q, NH) = dist(p_1, q)$. The points p_1, p_2, p_3 already finds q as their nearest facility. Therefore, this NH is a RNNH 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. Moreover, the time complexity of this enumeration is exponential in terms

of the number of points in P, although the checking of the neighbourhood distances is linear in terms of time complexity as the nearest facility of each point in the neighbourhood can be computed offline. Therefore, the naïve algorithm is inefficient and is not a viable solution for RNNH query processing.

2) AN EFFICIENT ALGORITHM

In order to overcome the issue associated with the naïve RNNH query processing algorithm, we propose an R-tree based algorithm to achieve better performance. The algorithm first computes a Voronoi diagram [8] of the query facility q, which is generated based on the perpendicular bisectors of the line segments $\overline{q,f}$ between the facilities $f \in F$ and q after indexing the facility dataset F into an R-tree and then, retrieving the facilities $f \in F$ from the R-tree in order of their distances to q. The perpendicular bisector of the line segment $\overline{q,f}$, denoted by $B_{q:f}$, divides the data space into two half-planes. Assume $H_{q:f}$ denotes the half-plane that contains the query facility q. The Voronoi diagram Z_q is the intersection of all of these half-planes $H_{q:f}$ w.r.t. all facilities $f \in F$ and q. All points located inside the Voronoi diagram of q finds q nearer than any other facility $f \in F$. The algorithm then retrieves all points $p \in P$ that are located inside the Voronoi diagram of the query and the points list is denoted by Z_p.

Consider the dataset of facilities, points and the query given in Figure 3. The Voronoi diagram Z_q of the query facility q is illustrated in Figure 5 containing $Z_p = \{p_1, p_2, p_3, p_{10}, p_{11}, p_{17}\}$. All of these points find the query containing q nearer than any other facility in $\{A, B, \ldots, E\}$.

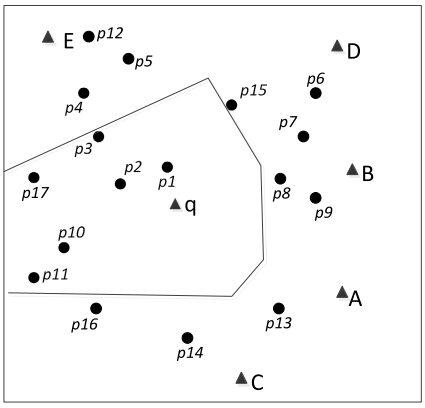


FIGURE 5. Voronoi cell of the query facility q.

Upon creating the Z_q for the query facility q and to retrieve the corresponding Z_p, the algorithm begins to generate the first neighbourhood by retrieving the nearest point of q inside the Z_q, called *firstclose*[1] by placing the points in Z_p into a min heap. The algorithm also calculates $d_H(q, NH)$ which is the distance between q and the *firstclose*. Next, the algorithm inserts the *firstclose* in a temporary list and retrieves all points from P close to the *firstclose* with a distance less than or equal to d (can be simply implemented as a range query after indexing the data points P into an R-tree). The distance of each of these points to its nearest facility is determined (nearest facility of each point $p \in P$ could be precomputed by the server to speed up the algorithm). If the nearest facility is not q and its distance to its nearest facility is not greater than $d_H(q, NH)$ and its distance to the current NH is less than its distance to its nearest facility, it is added to the list, otherwise, it is discarded. The point *firstclose* in the list is marked as visited. this process is repeated for all unvisited points in the list and the algorithm stops building 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 of q[2]; otherwise, it is discarded. The algorithm then retrieves the next *firstclose* from Z_p that has not been included in any previous neighbourhood and continues to find the corresponding neighbourhood for it. The above neighbourhood making process is continued until Z_p becomes empty.

III. RELATED WORK

Moving objects in spatial queries have been studied extensively in recent works such as [9], [10] and [11]. Since the continuous monitoring of moving queries has a significant role in studies related to spatial databases, many have investigated the monitoring of moving objects such as [12], [13] and [14]. One of the most fundamental algorithms used to examine moving objects refers to RNN [15]. The RNN has been studied extensively in the literature such as [16]–[26] since its introduction by [27]. The first work that presented the moving RNN was [28], which assumed the objects' speeds to be known. Many studies have extended this approach, but did not look into the motion patterns of the object [29]. [30] and [31] were the first to address continuous RNN based on six regions and a TPL algorithm has been proposed without assuming objects' speeds. The solution given in [30] is based on the approach of the six 60° regions to monitor RNN query results, while [31] proposed a monitoring algorithm called TPL for moving RNN queries based on a bisector approach.

Moving objects slash communication and computation costs through the concept of safe region (SR). In an SR, a query point does not need to be connected to the server. It is required to update its location only upon leaving the SR. Several studies have identified various types of SRs such as [32] and [33]. [33] presents the SR-based approach to monitor the moving skyline queries in Euclidean space, while [34] the SR concept to road networks for finding safest paths. Additionally, [35] addressed the SR concept by reporting the query locations to the server after t time and d distance. The study assumed d and t units to parameterise the SR, thereby reducing the communication overhead between moving clients and server. [36] addressed the SR concept, but limited it to a rectangular SR that is not applicable for a moving,

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

[2]It is possible to have more than one *firstclose* in the same reverse nearest neighbourhood (RNNH) of a query facility q.

circular SR. Although the first work to address the circular SR is [37], it omitted the calculation of area, as the server was in stand-by mode at a certain distance. Finally, [38] investigated circular SR for range query using the Monte-Carlo method, mainly due to the irregular shape of SR. However, the method can be applied only to range moving queries.

There also exist several algorithms in the literature that can be used to establish P2P communication in a mobile environment; these include lowest ID [39], largest-connectivity [40] and mobility-based clustering algorithms [41]. Many have also investigated query processing in the context of P2P. The first demand for a P2P algorithm was in regard to NN queries introduced in [42]. More works are notable for mobile P2P systems in spatial databases such as [43] that proposed P2PRdNN, albeit with a limited focus on monochromatic queries.

Next, [4] extended the reverse nearest neighbour (RNN) query by proposing reverse nearest neighbourhood query (RNNH) in spatial databases in a bichromatic manner. Instead of returning the dispersed RNN users that find the query as their closest facility, a method was proposed to find the location of the nearest group of users by considering the query point as the facility nearest to the group. Various devices can be managed in a P2P environment via a bichromatic approach. This study differs from previous investigations, as it is the first to assess bichromatic RNNH queries in mobile P2P environments by addressing the SR concept for RNNH queries.

IV. SAFE REGION FOR MOVING RNNH QUERIES

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 maximise 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 2), the decision regarding safe region to employ 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 for authorities in managing this kind of disaster.

A. 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 RNNH. 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 RNNH of the query q, while *firstclose* refers to the closest point of the query q in the RNNH. 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 RNNH.

Let us consider the example given in Figure 6, which shows the RNNH result for the query q, i.e., $NH = \{p_1, p_2, p_3, p_4\}$, where the neighbourhood points are connected with solid lines. In Figure 6, 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) and eliminates the need for re-computation 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.

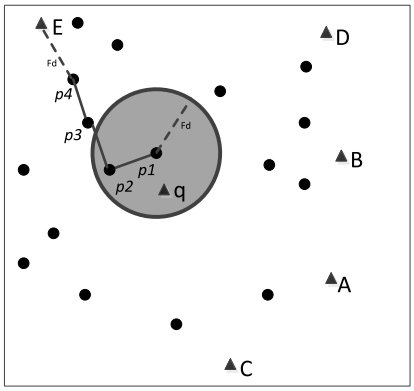


FIGURE 6. Basic safe region.

When the query q moves out of the shaded area in Figure 6, 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 7. If the query point moves out in any direction within the shadowed area as shown in Figure 6, 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 6). 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

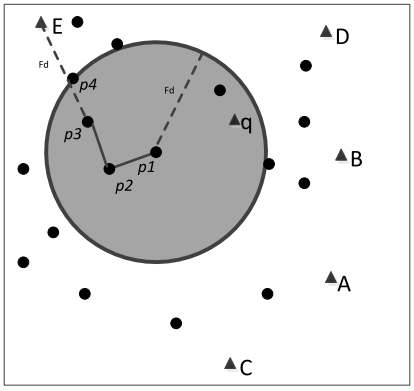


FIGURE 7. RNNH for the current location of q and the corresponding basic safe region.

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 1: 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.

Proof: (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 2: An object is discarded from the moving query RNNH result if its distance from a competitor facility is less than $dist(q, NH)$.

Proof: Assume that there are four points in the neighbourhood $NH = \{p_1, p_2, p_3, p_4\}$ (as illustrated Figure 8). 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 lower 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.

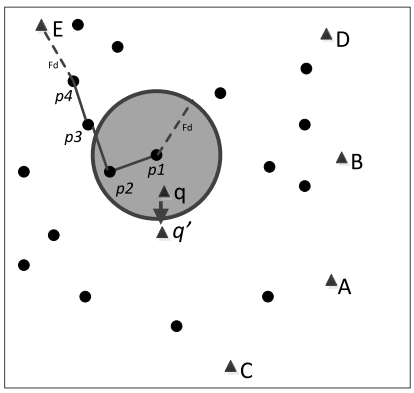


FIGURE 8. Query q moves to q': $dist(q, p_1) < F_d$ and $dist(q', p_1) > F_d$.

Lemma 3: 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: 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.

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

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

$$dist(p_1, q) - F_d = \begin{cases} > 0 & q \ \ outside \ \ the \ \ SR, \\ = 0 & q \ on \ the \ SR \ boundary, \\ < 0 & inside \ the \ SR \ boundary. \end{cases} \tag{3}$$

Based on the definition of RNNH query (*Definition* 4), $d_H(q, NH) \leq d_H(f, NH)$, $\forall p \in NH$ 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.

$$(q_x - p_{1x})^2 + (q_y - p_{1y})^2 - F_d^2 \leq 0 \tag{4}$$

If q lies outside the SR, then a recalculation of the RNNH is needed. Let us consider the example illustrated in Figure 8 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.

B. 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 1 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.

Lemma 4: 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: (By inspection) Figure 9, $(q_x; q_y)$ 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:

$$dist(q', q) = \sqrt{(q_x - q'_x)^2 + (q_y - q'_y)^2} \tag{5}$$

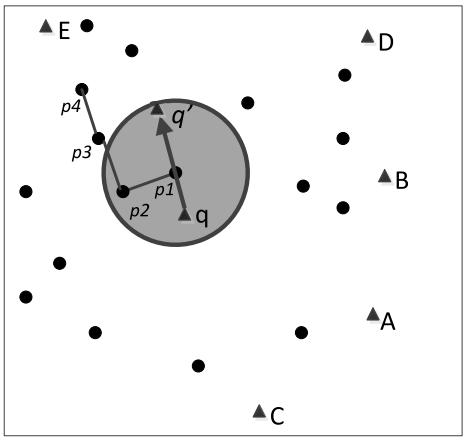


FIGURE 9. Lemma 4.

Based on Figure 9, we can conclude that $dist(q', q) > F_d$.

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 1 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 holds its objects of interest.

Lemma 5: 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: (By inspection). Based on Lemmas 2 and 4 we can conclude the following observations from the example displayed in Figure 10. 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 dismisses connection with server (see Figure 10).

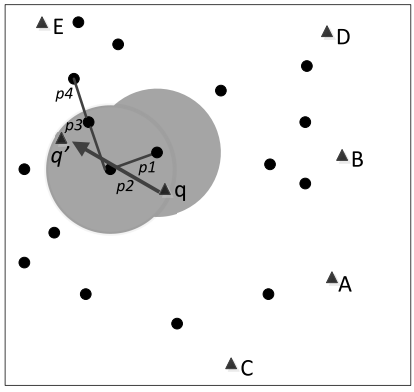


FIGURE 10. Enhanced safe region.

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 closest points 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 compared to the border of the Basic SR, as shown in Figure 10.

C. 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 *fristclose* point in the opposite direction, as stated in Lemma 1.

Lemma 6: 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.

Proof: 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 2, 4 and 5, we can conclude that query point q can move to the position of q' (as illustrated Figure 11), 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).

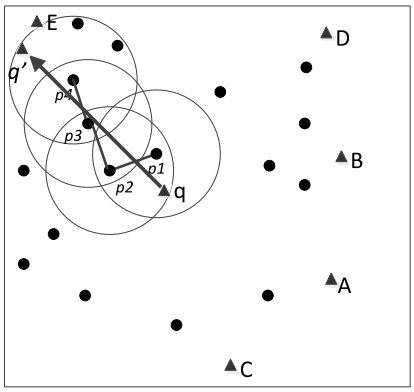


FIGURE 11. Query moves $2F_d$ distance.

Figure 12 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.

D. ALGORITHMS FOR SAFE REGIONS

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

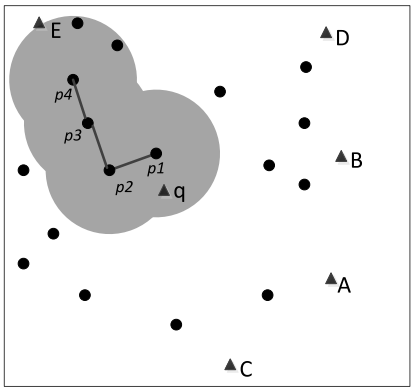


FIGURE 12. Extended safe region.

1) BASIC SR ALGORITHM

Algorithm 2 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 [4]. In line 2, Algorithm 2 calls the calculating boundary function. In algorithm 1, 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 Voronoi diagram or the Influence Zone introduced by [29]. 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. In line 12, 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 on for all SOL members, then the Calculate Boundary function returns boundary. Thus, Algorithm 2 receives the boundary and assign 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 a circle of range of the moving query, its radius is f_d from the nearest neighbour of q.

Algorithm 1 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;
```

Algorithm 2 Basic Safe Region Algorithm

```
Input: q: query point
Output: Basic safe region for q
SOL ← RNNH(d, m, q, U, F) ;
F_d ← CalculateBoundary(SOL) ;
Circle(p_i, F_d) ;
BasicSafeRegion ← Area(Circle) ;
```

2) ENHANCED SR ALGORITHM

Algorithm 3 is used to find the Enhanced SR for a moving query. Here, the SOL is calculated by using the RNNH algorithm [4]. In line 2, after calling Algorithm 1 for calculating the boundary, Algorithm 3 conducts the processing steps as we explained in Algorithm 2, and then should create the Enhanced SR based on the closest two objects of SOL to the query point. The area of Enhanced SR formed by two circles with a radius of F_d, whose centres are the two closest points to q. In this algorithm, q can move freely further away from the F_d distance from its current location.

Algorithm 3 Enhanced Safe Region Algorithm

```
Input: q: query point
Output: Enhanced safe region for q
SOL ← RNNH(d, m, q, U, F) ;
F_d ← CalculateBoundary(SOL) ;
Circle_1(p_i, F_d) ;
EnhancedSafeRegion ← Area(Circle_1) ;
Circle_2(p_{i+1}, F_d) ;
EnhancedSafeRegion ← Area(Circle_2) ;
```

3) EXTENDED SR ALGORITHM

Algorithm 4 calculates the Extended SR for a moving query. First, the set points of interest are found (RNNH result) [4] to SOL list. The Extended SR is created based on the steps shown in Algorithm 4. 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 4 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

algorithm, the query point can move more than 2 F_d to its current location.

V. CALCULATING THE AREA OF THE SAFE REGION

When an SR has one or two objects, the calculation of the area involves a straightforward geometric issue [38]. 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:

$$Rpi = \pi F_d^2 \tag{6}$$

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 $RP = \{Rp_1, Rp_2, \ldots, Rp_n\}$ of n circles, wherein F_d is the radius and the centres are $\{p_1, p_2, \ldots, p_n\}$. The circles in RP may partially overlap.

Figure 13 illustrates the formation of SRs by 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, $(A2, A3, \ldots, A10)$ 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.

$$dist(pi, q) \leq F_d \text{ and } Rpi \bigcap_{x \neq i} Rpx = \theta. \tag{7}$$

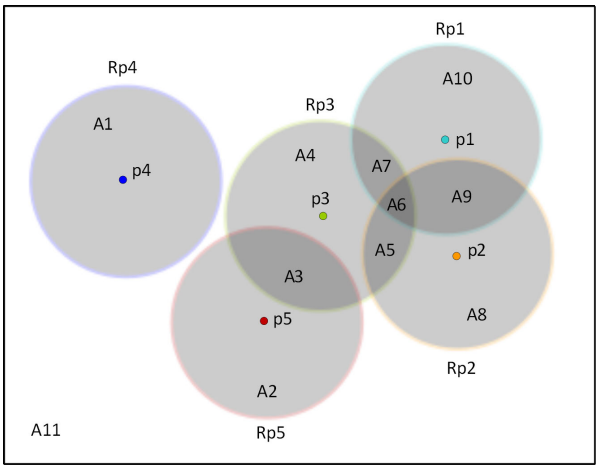


FIGURE 13. Types of extended safe regions.

A. CALCULATING THE TWO CIRCLES

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

$$dist(pi, q) \leq F_d \text{ and } Rpi \bigcap_{x \neq i} Rpx \neq \theta. \tag{8}$$

or

$$dist(pi, q) \leq F_d \text{ and } dist(pj, q) \leq F_d, \quad i \neq j. \tag{9}$$

$A2$, $A3$, and $A4$ (see Figure 13) 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 8, where d refers to the half distance between two points, and F_d is the area radius, as illustrated in Figure 14.

$$Rpi \cap Rpj = F_d^2 arccos(\frac{d}{F_d}) - d\sqrt{F_d^2 - d^2}. \tag{10}$$

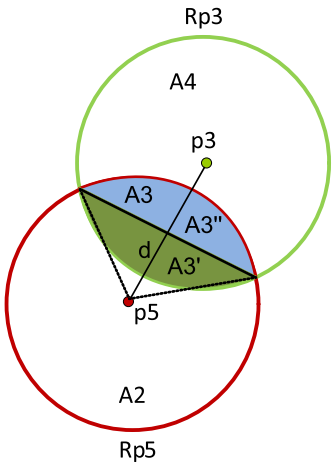


FIGURE 14. Area of intersection of two circles.

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

$$\text{Area of Safe Region} = Rpi + Rpj - (Rpi \cap Rpj). \tag{11}$$

B. 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 [44] can be applied to calculate the area. 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:

$$\frac{\text{sum of points in SR}}{\text{total random points}} \times \text{area of bounding region} \tag{12}$$

VI. 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 [45]. 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 2 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 3 presents the parameters applied for RNNH query processing algorithm (Section II-B) and the default values employed in the experiments.

TABLE 2. Datasets used in our experiments.

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

TABLE 3. 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%

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 (as a 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 represent various services offered by the service provider. Several query facilities (long-range devices) were randomly generated in low, medium and high-density environments.

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 the accuracy of the equation-based and simulation-based methods.

A. 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 V. 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. After obtaining the average of the calculation, it was compared with the results of the equation method to determine the accuracy of the simulation method in this study. Figure 15 illustrates the area of Basic SR that was calculated using the Monte-Carlo simulation and equation method. As shown in Figure 15, the results (exact areas) of the simulation method were very close to the results obtained by the equation method (Equation 6).

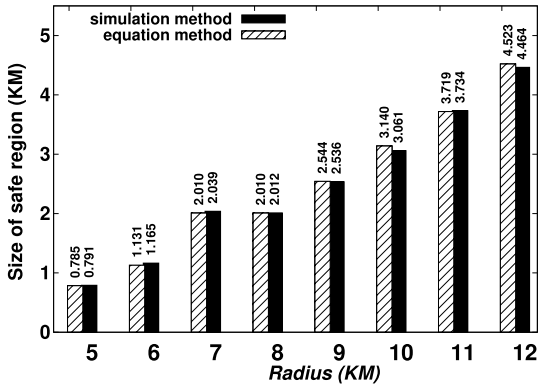


FIGURE 15. Simulation model vs. equation method.

Based on the experiments reported in this section, the bounding region was determined as 100 × 100 Km^2, with 100,000 random points used 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 16 displays a screen shot of the simulation software that was applied to calculate the SR areas to be compared with the whole area.

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 clearly shown in Figure 17.

B. 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 18 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

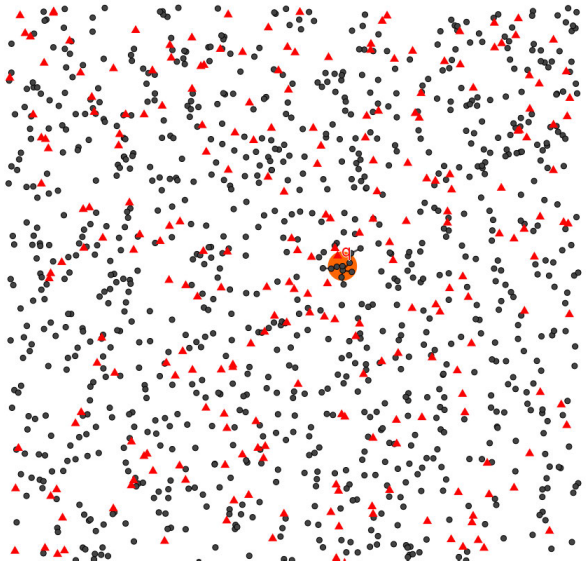


FIGURE 16. Demonstration of software in calculating the area of basic safe region corresponding to static query.

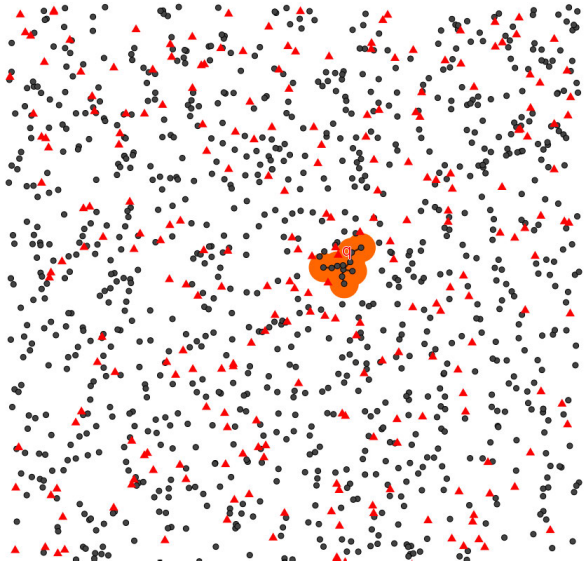


FIGURE 17. Demonstration of software calculating the area of extended safe region corresponding to static query.

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.

C. SIZE OF THE SAFE REGION

Figures 19 - 21 illustrate the comparison of the three types of SRs: Basic, Enhanced, and Extended, in three different environments (low: Figure 19, medium: Figure 20, and high: Figure 21) 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

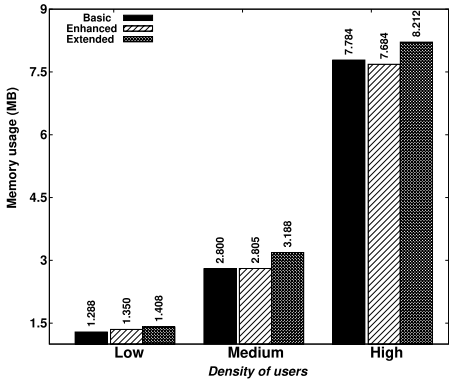


FIGURE 18. Memory usage.

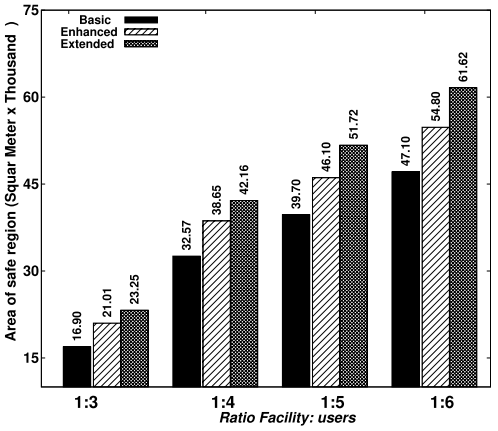


FIGURE 19. Low-density environment.

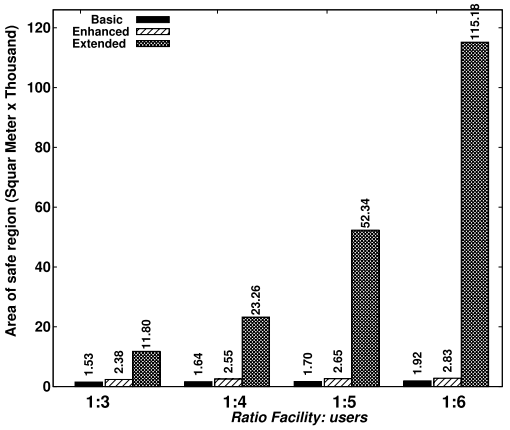


FIGURE 20. Medium-density environment.

the size of the Enhanced SR was greater than than that of the Basic SR (see Figure 19). 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 21). It was observed that the decrease in the number of facilities affected the SR size. This can be clearly seen in Figure 20, 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.

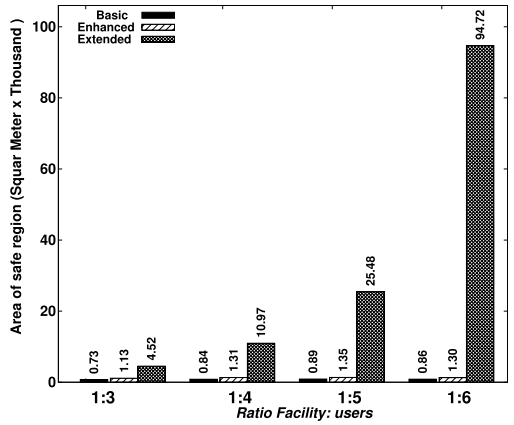


FIGURE 21. High-density environment.

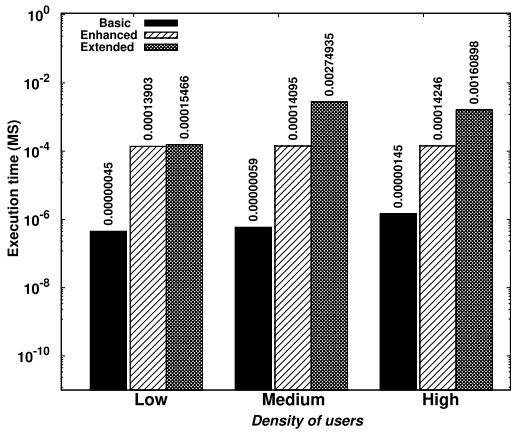


FIGURE 22. Construct CPU time needed for safe regions in three different density environments.

D. 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 22 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 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.

VII. CONCLUSION

This study proposes safe-region-based methods for moving RNNH queries in a peer-to-peer (P2P) environment. The objective of the SR is to determine a region where the corresponding query can move freely, while retaining an unchanged RNNH result. We proposed three safe-region based methods called *Basic*, *Enhanced* and *Extended* using geometric properties to maximise the SRs. The decision regarding the method to be employed 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 a higher computational time compared to other methods. However, while the *Basic* SR requires the lowest computational time, it offers the smallest SR area.

REFERENCES

[1] T. P. Nghiem, K. Maulana, K. Nguyen, D. Green, A. B. Waluyo, and D. Taniar, "Peer-to-peer bichromatic reverse nearest neighbours in mobile ad-hoc networks," *J. Parallel Distrib. Comput.*, vol. 74, no. 11, pp. 3128–3140, Nov. 2014.

[2] O. Al-Bayari and B. Sadoun, "New centralized automatic vehicle location communications software system under GIS environment," *Int. J. Commun. Syst.*, vol. 18, no. 9, pp. 833–846, Nov. 2005.

[3] A. Hameurlain, J. Küng, and R. Wagner, *Transactions on Large-Scale Data-and Knowledge-Centered Systems I*, vol. 5740. Berlin, Germany: Springer, 2009.

[4] N. Allheeib, M. S. Islam, D. Taniar, Z. Shao, and M. A. Cheema, "Density-based reverse nearest neighbourhood search in spatial databases," *J. Ambient Intell. Humanized Comput.*, vol. 2018, pp. 1–12, Oct. 2018.

[5] H.-C. Lee and K.-H. Ke, "Monitoring of large-area IoT sensors using a LoRa wireless mesh network system: Design and evaluation," *IEEE Trans. Instrum. Meas.*, vol. 67, no. 9, pp. 2177–2187, Sep. 2018.

[6] K. E. Potts, R. M. Bennett, and A. Rajabifard, "Spatially enabled bushfire recovery," *GeoJournal*, vol. 78, no. 1, pp. 151–163, Feb. 2013.

[7] R. Blanchi, J. Leonard, K. Haynes, K. Opie, M. James, and F. D. D. Oliveira, "Environmental circumstances surrounding bushfire fatalities in Australia 1901–2011," *Environ. Sci. Policy*, vol. 37, pp. 192–203, Mar. 2014.

[8] K. Xuan, G. Zhao, D. Taniar, W. Rahayu, M. Safar, and B. Srinivasan, "Voronoi-based range and continuous range Query processing in mobile databases," *J. Comput. Syst. Sci.*, vol. 77, no. 4, pp. 637–651, Jul. 2011.

[9] Z. Deng, M. Wang, L. Wang, X. Huang, W. Han, J. Chu, and A. Zomaya, "An efficient indexing approach for continuous spatial approximate keyword queries over Geo-textual streaming data," *ISPRS Int. J. Geo-Inf.*, vol. 8, no. 2, p. 57, 2019.

[10] S. Alamri, D. Taniar, and M. Safar, "A taxonomy for moving object queries in spatial databases," *Future Gener. Comput. Syst.*, vol. 37, pp. 232–242, Jul. 2014.

[11] C. Salgado, M. A. Cheema, and M. E. Ali, "Continuous monitoring of range spatial keyword Query over moving objects," *World Wide Web*, vol. 21, no. 3, pp. 687–712, May 2018.

[12] H. AL-Khalidi, D. Taniar, and M. Safar, "Approximate algorithms for static and continuous range queries in mobile navigation," *Computing*, vol. 95, nos. 10–11, pp. 949–976, Oct. 2013.

[13] T. Ma, J. Jia, Y. Xue, Y. Tian, A. Al-Dhelaan, and M. Al-Rodhaan, "Protection of location privacy for moving KNN queries in social networks," *Appl. Soft Comput.*, vol. 66, pp. 525–532, May 2018.

[14] M. Zhang, N. Rishe, L. Weitong, J. Lazarre, and T. Li, "Voronoi diagram-based algorithm for efficient progressive continuous *k*-nearest neighbor Query for moving objects," U.S. Patent 10 200 814, Feb. 5, 2019.

[15] D. Taniar and W. Rahayu, "A taxonomy for region queries in spatial databases," *J. Comput. Syst. Sci.*, vol. 81, no. 8, pp. 1508–1531, Dec. 2015.

[16] M. A. Cheema, X. Lin, W. Wang, W. Zhang, and J. Pei, "Probabilistic reverse nearest neighbor queries on uncertain data," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 4, pp. 550–564, Apr. 2010.

[17] Q. T. Tran, D. Taniar, and M. Safar, "Bichromatic reverse nearest-neighbor search in mobile systems," *IEEE Syst. J.*, vol. 4, no. 2, pp. 230–242, Jun. 2010.

[18] I. Stanoi, D. Agrawal, and A. El Abbadi, "Reverse nearest neighbor queries for dynamic databases," in *Proc. ACM SIGMOD Workshop Res. Issues Data Mining Knowl. Discovery*, 2000, pp. 44–53.

[19] Q. T. Tran, D. Taniar, and M. Safar, "Reverse *k* nearest neighbor and reverse farthest neighbor search on spatial networks," in *Transactions on Large-Scale Data- and Knowledge-Centered Systems I*. Berlin, Germany: Springer, 2009, pp. 353–372.

[20] Y. Tao, D. Papadias, and X. Lian, "Reverse KNN search in arbitrary dimensionality," *Proc. VLDB Endowment*, vol. 30, pp. 744–755, Aug. 2004.

[21] E. Achtert, H.-P. Kriegel, P. Kröger, M. Renz, and A. Züfle, "Reverse k-nearest neighbor search in dynamic and general metric databases," in *Proc. 12th Int. Conf. Extending Database Technol. Adv. Database Technol. (EDBT)*, 2009, pp. 886–897.
[22] I. Stanoi, M. Riedewald, D. Agrawal, and A. El Abbadi, "Discovery of influence sets in frequently updated databases," *Proc. VLDB*, vol. 2001, pp. 99–108, Sep. 2001.
[23] M. A. Cheema, W. Zhang, X. Lin, Y. Zhang, and X. Li, "Continuous reverse k nearest neighbors queries in Euclidean space and in spatial networks," *VLDB J.*, vol. 21, no. 1, pp. 69–95, Feb. 2012.
[24] M. A. Cheema, X. Lin, Y. Zhang, W. Wang, and W. Zhang, "Lazy updates: An efficient technique to continuously monitoring reverse KNN," *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 1138–1149, Aug. 2009.
[25] T. Bernecker, T. Emrich, H.-P. Kriegel, M. Renz, S. Zankl, and A. Züfle, "Efficient probabilistic reverse nearest neighbor Query processing on uncertain data," *Proc. VLDB Endowment*, vol. 4, no. 10, pp. 669–680, Jul. 2011.
[26] S. Yang, M. A. Cheema, X. Lin, and Y. Zhang, "SLICE: Reviving regions-based pruning for reverse k nearest neighbors queries," in *Proc. IEEE 30th Int. Conf. Data Eng.*, Mar. 2014, pp. 760–771.
[27] F. Korn and S. Muthukrishnan, "Influence sets based on reverse nearest neighbor queries," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 201–212, Jun. 2000.
[28] R. Benetis, C. S. Jensen, G. Karciauskas, and S. Saltenis, "Nearest neighbor and reverse nearest neighbor queries for moving objects," in *Proc. Int. Database Eng. Appl. Symp.*, 2002, pp. 44–53.
[29] M. A. Cheema, X. Lin, W. Zhang, and Y. Zhang, "Influence zone: Efficiently processing reverse k nearest neighbors queries," in *Proc. IEEE 27th Int. Conf. Data Eng.*, Apr. 2011, pp. 577–588.
[30] T. Xia and D. Zhang, "Continuous reverse nearest neighbor monitoring," in *Proc. 22nd Int. Conf. Data Eng. (ICDE)*, 2006, p. 77.
[31] J. M. Kang, M. F. Mokbel, S. Shekhar, T. Xia, and D. Zhang, "Continuous evaluation of monochromatic and bichromatic reverse nearest neighbors," in *Proc. IEEE 23rd Int. Conf. Data Eng.*, Apr. 2007, pp. 806–815.
[32] S. Oh, H. Jung, and U.-M. Kim, "An efficient processing of range spatial keyword queries over moving objects," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2018, pp. 525–530.
[33] M. A. Cheema, X. Lin, W. Zhang, and Y. Zhang, "A safe zone based approach for monitoring moving skyline queries," in *Proc. 16th Int. Conf. Extending Database Technol. (EDBT)*, 2013, pp. 275–286.
[34] S. Aljubayrin, J. Qi, C. S. Jensen, R. Zhang, Z. He, and Z. Wen, "The safest path via safe zones," in *Proc. IEEE 31st Int. Conf. Data Eng.*, Apr. 2015, pp. 531–542.
[35] D. Yung, M. L. Yiu, and E. Lo, "A safe-exit approach for efficient network-based moving range queries," *Data Knowl. Eng.*, vol. 72, pp. 126–147, Feb. 2012.
[36] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee, "Location-based spatial queries," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 2003, pp. 443–454.
[37] M. A. Cheema, L. Brankovic, X. Lin, W. Zhang, and W. Wang, "Continuous monitoring of distance-based range queries," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 8, pp. 1182–1199, Aug. 2011.
[38] H. AL-Khalidi, D. Taniar, J. Betts, and S. Alamri, "On finding safe regions for moving range queries," *Math. Comput. Model.*, vol. 58, nos. 5–6, pp. 1449–1458, Sep. 2013.
[39] A. Ephremides, J. E. Wieselthier, and D. J. Baker, "A design concept for reliable mobile radio networks with frequency hopping signaling," *Proc. IEEE*, vol. 75, no. 1, pp. 56–73, 1987.
[40] A. K. Parekh, "Selecting routers in ad-hoc wireless networks," in *Proc. SBT/IEEE Int. Telecommun. Symp.*, Rio de Janeiro, Brazil, vol. 204, Aug. 1994, pp. 5–9.
[41] G. H. K. Lam, H. V. Leong, and S. C. F. Chan, "GBL: Group-based location updating in mobile environment," in *Database Systems for Advanced Applications*. Berlin, Germany: Springer, 2004, pp. 762–774.
[42] W.-S. Ku and R. Zimmermann, "Nearest neighbor queries with peer-to-peer data sharing in mobile environments," *Pervas. Mobile Comput.*, vol. 4, no. 5, pp. 775–788, Oct. 2008.
[43] D. Chen, J. Zhou, and J. Le, "Reverse nearest neighbor search in peer-to-peer systems," in *Flexible Query Answering Systems*. Berlin, Germany: Springer, 2006, pp. 87–96.
[44] B. D. Ripley, *Stochastic Simulation*, vol. 316. John Hoboken, NJ, USA: Wiley, 2009.
[45] T. Brinkhoff, "A framework for generating network-based moving objects," *GeoInformatica*, vol. 6, no. 2, pp. 153–180, 2002.

NASSER ALLHEEIB received the bachelor's degree in computer science from King Saud University, Saudi Arabia, in 2008, and the master's degree in information technology from the School of Information Technology, Rochester Institution Technology, New York, USA, in 2014. He is currently pursuing the Ph.D. degree with the Clayton School of Information Technology, Monash University, Australia. His research interests include static and moving objects databases, query processing, and spatial databases.

DAVID TANIAR received the bachelor's, master's, and Ph.D. degrees in computer science, with a particular specialty in databases. He is currently an Associate Professor with the Faculty of Information Technology, Monash University, Australia. He has published a book *High Performance Parallel Database Processing and Grid Databases* (John Wiley and Sons, 2008). He has published over 130 research articles that can viewed at the DBLP server. His current research interests include spatial query processing and parallel databases. He is the Founding Editor-in-Chief of *International Journal of Data Warehousing and Mining, International Journal of Web and Grid Services*, and *International Journal of Web Information Systems*.

HAIDAR AL-KHALIDI received the Ph.D. degree from Monash University, in 2010. He is currently an Engineering (IT) Subject Leader with the Monash College, Australia. He has published more than 15 articles in high ranked journals and presented at international conferences. He received an Australian Postgraduate Awards scholarship APA towards finishing his Ph.D. at Monash University. His list of publications can be viewed at the DBLP server (http://dblp.uni-trier.de/pers/hd/a/Al=Khalidi:Haidar).

MD. SAIFUL ISLAM received the B.Sc. (Hons.) and M.S. degrees in computer science and engineering from the University of Dhaka, Bangladesh, in 2005 and 2007, respectively, and the Ph.D. degree in computer science and software engineering from the Swinburne University of Technology, Australia, in February 2014. He is currently a Lecturer with the School of Information and Communication Technology, Griffith University, Australia. His current research interests are in the areas of database usability, spatial data management, and big data analytics.

KIKI MAULANA ADHINUGRAHA received the bachelor's, master's, and Ph.D. degrees in information technology. He is currently a Research Fellow with the School of Engineering and Mathematical Science, La Trobe University, Melbourne, Australia. His research interests include database management, database architecture, spatial query processing, and computational geometry.