CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. (2013)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3056

Finding reverse nearest neighbors by region

Kiki Maulana Adhinugraha[*,†], David Taniar and Maria Indrawan

Faculty of Information Technology, Monash University, Melbourne, Australia

SUMMARY

Common reverse nearest neighbor queries in spatial database run in an inefficient way because they need to check a query result with almost every nearest neighbor. This wastes many time and resources, making this approach unsuitable for mobile computation. Instead of using the neighbors as candidates for the query result, a region approach can be used to answer the query. By using this approach, any objects located in the region will be considered candidate results for the query. To reduce the cost of creating the region, we introduce the concept of a contact zone, a method that can identify the right region generator points without having to process the whole points in the space, hence make reverse nearest neighbor queries by region possible to be run in mobile devices. Copyright © 2013 John Wiley & Sons, Ltd.

Received 29 April 2013; Revised 1 June 2013; Accepted 6 May 2013

KEY WORDS: RNN region; influence zone; contact zone; peers; Voronoi cell; Voronoi diagram; region vertex

1. INTRODUCTION

Mobile computing has been a common activity nowadays, and research shows that the usage of mobile devices increases significantly. Because almost all recent mobile devices have been equipped with better CPU, larger storage, faster network speed, and Global Positioning System (GPS) navigation system as the standard features, mobile devices will be able to do some computing tasks where these tasks used to be carried out in common computer systems [1]. One of the most common tasks that can be carried out in mobile device is navigation. The example of basic navigation is to find the route from points A to B, where these tasks only need the map that has been stored in the mobile's storage or can be accessed online. The more advanced navigation task is to guide the route from point A to point B that can be carried out by using GPS module in mobile's system or to find a point of interests in a specific area that also can be performed by applying k-nearest neighbor(k-NN) algorithm in the mobile's navigation system [2–8].

Although a mobile device can be used to find k-NN objects, no algorithm has been developed to find reverse nearest neighbor (RNN) for mobile devices. This is because the nature of RNN where each object has to be verified to the query point before the object can be acknowledged as the result. Because the RNN query needs high computation resource, this type of query is not suitable for mobile environment [9].

In this paper, we propose a new method to answer RNN query by region that only needs to process some points to obtain the answer; hence, this method is suitable for mobile device environment. Our experiments show that this method can create RNN region from only a few points, and this query can be generated from any points in the space.

*Correspondence to: Kiki Maulana Adhinugraha, Faculty of Information Technology, Monash University, Melbourne, Australia.
†E-mail: kiki.adhinugraha@monash.edu

The paper is organized as follows. Section 2 provides the motivation and problem definition, Section 3 describes detailed region generation using the contact zone, and Section 4 describes abut evaluation. Section 5 discusses about related works, and Section 6 is the conclusion about the paper.

2. PROBLEM DEFINITION

The RNN concept that had been proposed more than a decade ago had attracted many researchers attention, especially in spatial databases [10], because of its capability in answering the query in objects' perspective rather than from the query point [11]. However, because the common method to answer the RNN query uses point-to-point approach [12–16], where each object needs to be verified if the query point is considered as the nearest, answering RNN query will take more efforts rather than common k-NN query. Instead of using point-to-point approach, region approach [17] can be used to avoid object verification phase, because the query point has its influence zone [18], and every object within this influence zone will always consider query point as the nearest. According to previous research, the shape of the influence zone for each point is always convex polygon, and this polygon is actually a Voronoi cell where $k = 1$ [19]. The union of the whole Voronoi cell will construct a Voronoi diagram [20].

Because of its unique characteristic, Voronoi diagram has commonly been used to solve spatial queries [21], which can be used to solve k-NN [6, 22] or RNN [18, 21, 23] because of its characteristic that can identify the region of the generator points.

Figure 1 shows the relation between Voronoi diagram and reverse nearest objects from query point q. As shown in the figure, there are three objects $0_1, o_2$, and o_8 in the Voronoi cell of q, where these objects consider q as the nearest point. Any other objects located outside the Voronoi cell of q will not consider q as the nearest point; instead, they will consider the generator point of Voronoi cell where they are located as the nearest point.

Because the region of a query point is influenced by another points in the space, we may need to include some points in the process of region generation. The points that influence the creation of the query point's region is called the region generator points, whereas the points that have no influence to the query point's region is called unused points. Not all points in the space can be the region generator points. The most difficult part in creating the region of a query point is to find the right region generator points. Some works had been performed in identifying the region [17, 18]; however, because these methods are based on candidate region, these methods will cause a scenario where some unused points are left in the candidate region and need another refinement process.

In this paper, we will introduce contact zone, a new method in identifying the region generator points that will construct the influence zone of a query point. Any point in the space can be the query point, and our method can be applied in any query point position. The contact zone CZ is the circle $C(v_j, \text{dist}(v_j, q))$ in a vertex v_j where any points within this zone will affect the influence zone.

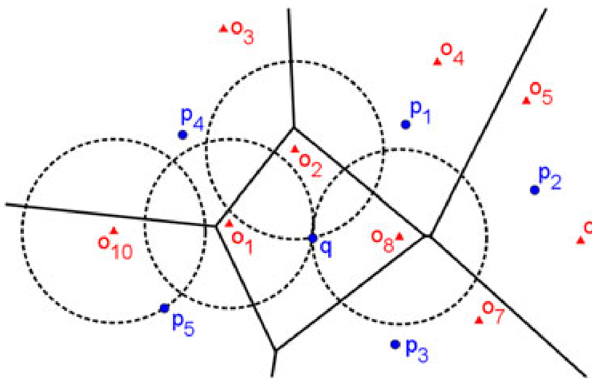
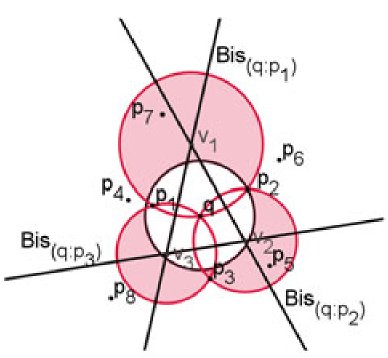


Figure 1. Reverse nearest objects in Voronoi diagram.

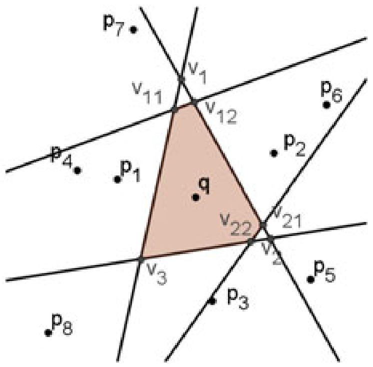
The vertex v_i is constructed from the intersection of two or more bisector lines from query point q to any other points. Because any points can be query point q, the rest of the points will be called as peers $P = p_1, p_2, p_3, \ldots p_n$.

Figure 2(a) shows the example of a contact zone (the shaded area) in an initial region. The query point q is surrounded by peers p_1 to p_8 where these peers are sorted on the basis of the distance $\text{dist}(q, p_i)$. The initial region is constructed by processing each peers from the nearest until the polygon is created. After the initial region is created, contact zones are defined on each vertex, and this method only processes all peers inside the contact zones and excludes all peers outside the contact zone. In this case, peers p_4, p_6, and p_8 will be excluded, and $p_5$andp_7 will be processed, even though $\text{dist}(q, p_4) < \text{dist}(q, p_7)$. Peer p_5 has a bisector line that will cut through the initial region in vertex v_2, whereas peer p_2 has a bisector line that will cut through the initial region in vertex v_1. Meanwhile, peers p_4, p_6, and p_8 do not have a bisector line than can affect the initial region Figure 2(b) shows the RNN region for query point q after p_5 and p_7 are processed. As shown in the figure, the RNN region is constructed from peers p_1, p_2, p_3, p_5, and p_7, whereas other peers were ignored.

Let site $S = s_1, s_2, \ldots s_n$ be the point in Euclidean space that has influence, where object $O = o_1, o_2, \ldots o_m$ is the object of interest (Table I). Query point q is the site that issues the query, whereas other sites will be consider as peers $P = p_1, p_2, \ldots, p_n$. RNN query from a query point is a method to find all objects that consider the query point q as the nearest point rather than to any other peers. To obtain the desired answer, the distance of each object o_i to query point must be



(a) Contact Zone with Initial Region



(b) RNN Region

Figure 2. Contact zone in determining the next peers to be processed. (a) Contact zone with initial region and (b) reverse nearest neighbor (RNN) region.

Table I. List of notation.

Notation	Definition
$S = \{s_1, s_2, \ldots, s_n\}$	Sites
$q \in S$	Query point
$P = \{p_1, p_2, \ldots, p_n\}$	Peers
$P = S - \{q\}$	
$O = \{o_1, o_2, \ldots, o_m\}$	Object of interest from the query point
$\text{Bis}(q : p_i)$	Bisector line between q and p_i
$V = \{v_1, v_2, \ldots, v_x\}$	Vertex
$\text{RV} = \{rv_1, rv_2, \ldots, rv_y\}$	Region vertex, all vertices that become the vertex of
$\text{RV} \in V$	RNN region
Sp_i	Half-plane of p_i that is created by $\text{Bis}(q : p_i)$
Sq_i	Half-space of q that is created by $\text{Bis}(q : p_i)$
$\text{Poly}(v_1, v_2, \ldots, v_x)$	Polygon with vertex point $v1, v2, \ldots, v_x$

less than the distance from the object to any other peers. This means that the more objects or peers exist in the space, the more verification needs to be carried out for the whole objects and peers. This approach is usually called the point-to-point approach. There are some problems that might exist in the point-to-point approach, which are

1. Each object's distance from q and other peers needs to be checked. When the number of objects or peers increases, the number of processing the distance will also increase.
2. When the objects or the sites move, the process of distance comparing needs to be redone.

To overcome this matter, the region approach is used. Instead of using the objects as the basis, the region approach uses the query point and peers as the basis, considering query point and peers as sites, where each site has its own influence zone.

Definition 1
The RNN region of a query point $RNN(q)$ is an area surrounded by the boundary in convex polygon shape, where every object O located in this region will always consider query point q as the nearest point. RNN region is also called influence zone or Voronoi cell where $k = 1$.

Definition 2
Initial region is the first polygon created by intersecting a bisector line $\mathrm{Bis}(q, p_i)$ from the nearest peer until the polygon is created.

Because the RNN region is a Voronoi cell, the Voronoi diagram is needed to retrieve this region, and the Voronoi diagram can be created after the whole sites have been processed. This means to have a single region from the query point q, all peers P have to be processed, and because not all peers are necessary, it is not efficient to process the peers that have no effect in region creation. Contact zone is a method in determining the peers that have to be processed and excluded. The contact zone is based on the limited area around the vertex where any peers inside this area will always have an effect to the vertex. By using the contact zone, only peers that are really needed in region creation are selected, and the unnecessary peers will be excluded.

3. RELATED WORKS

The RNN is a method to retrieve all objects that consider the query point as the nearest neighbor. This problem can be solved by using point-to-point approach or by region-based approach. Point-to-point approach is a method where each object is processed to compare the distance from the object to query point and from the object to other query point rivals. However, this method is not suitable when there are too many objects in the space.

Instead of trying each possible objects and calculating their distance to the query point, some people focused their research in creating the index for RNN query [24, 25]. Congjun *et al.* [24] created an RdNN-tree (R*-tree with the distance of nearest neighbor) index to store the point location and enhance each nodes with information about the distance of the nearest neighbors of the points in the nodes. Unlike the RNN-tree [11] that does not store the objects in the database, the RdNN-tree extends the RNN-tree by storing the objects of the database itself in an R-tree rather than circle around them. To answer this problem, Achtert *et al.*[26] proposed MRkNNCoP-Tree, which is similar to RdNN-tree but stores approximation all k-NN distances for any data object rather than the exact k-NN distance for one fixed k. The only limitation is that k is constrained by a parameter $k_{\max}$ specifying the maximal value of k that can be supported. However, another indexing method is needed in mobile devices environment, where the objects or the query points might move frequently [27–29].

Another approach in resolving the RNN query is by using the region approach. From previous research, the RNN region is a Voronoi cell where $k = 1$ [19]. Although a Voronoi diagram is used for common spatial nearest neighbor problems [30–33], some researchers also use it to solve the RNN problem [9, 21, 34].

Although creating the whole Voronoi diagram is not effective, because only one Voronoi cell will be used to answer the query, some researchers try to create this Voronoi cell with various approximation approach [17, 18]. Wu *et al.* [17] proposed INCH method based on candidate region, where the bisector for each data point must be computed to create this candidate region. The next refinement process is to select the right region generator points. Because this method must process each data point, this method cannot be used without using the whole points in the space. Similar to this method, Cheema *et al.* [18] proposed a method to generate the influence zone by using candidate region and followed by refinement process. Because initial region is the whole space, again this method is not suitable for mobile computation where the number of points is limited.

In this paper, we introduce a new method in determining the region generator points without having to compute the whole points in the space by using contact zone. We will show the application of contact zone in two different cases, the first one is the application of contact zone with candidate region or initial region, and the second one is the application of contact zone without the initial region. Our experiments show that the application of contact zone without initial region only requires less points to be computed compared with contact zone with initial region.

4. REGION CREATION

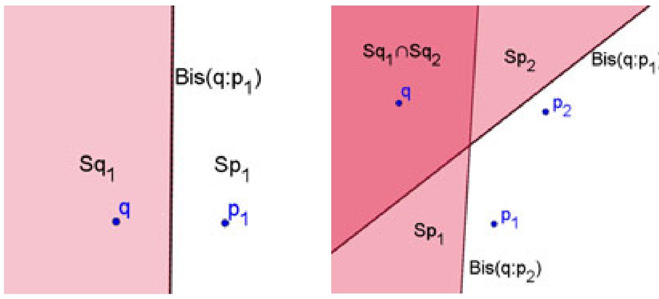
The RNN query by region approach is based on the concept where the query point q has its own region called influence zone, where in this influence zone, all objects o_i will consider the query point q as the nearest point, compared with any other peers. The influence zone is in a convex polygon shape, and it is actually a Voronoi cell, where this cell can be obtained from the whole Voronoi diagram. The Voronoi diagram is the union of the whole Voronoi cell where each site will have their own region, and no regions are overlapped with others. The region for each generator point is based on a bisector line, where the bisector line divides the area into two equal sizes called half-space.

Definition 3
A bisector line $\text{Bis}(q : p_i)$ is a line between p_i and q that divides the space into two equal size areas Sp_i and Sq_i. Sp_i is called a half-plane of p_i, and Sq_i is a half-plane of q from peer p_i.

The region for RNN(q) or Voronoi cell is defined as the intersection of the whole half-space Sq_i from whole peers P.

$$\text{RNN}(q) = \bigcap_{i=1}^{n} Sq_i$$

Figure 3(a) shows the definition of the region that is generated from two points, where the region for one point is a half-space that is created from the bisector line. In Figure 3(b), the region of q is created by the intersection of all the half-spaces of q.



(a) Region from two points (b) Region from three points

Figure 3. Region definition with bisector line. (a) Region from two points and (b) region from three points.

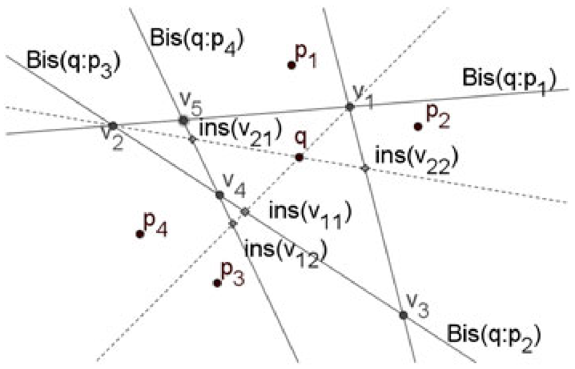


Figure 4. Vertex and region vertex.

Definition 4
Vertex $V = \{v_1, v_2, \ldots, v_x\}$ is the intersection between two or more bisector lines $\mathrm{Bis}(q : p_i)$.

Definition 5
Region vertex $\mathrm{RV} = \{rv_1, rv_2, \ldots, rv_x\}$ is the vertex of the region RNN(q), RV $\in V$.

Lemma 1
Let $lv(q, v_j)$ be the line that is formed from point q. A vertex v_i is also a region vertex if $lv(q, v_j$ does not have an intersection with any available bisector line $\mathrm{Bis}(q : p_i)$ or the intersection is not located between q and v_j.

Proof
Figure 4 shows the region of q that is created from four peers. The peers are sorted by the distance from the nearest to the farthest. uPolygon Poly(v_1, v_2, v_3) is the initial region of RNN(q). When p_4 is processed, its bisector line cuts through the initial region and creates new vertices v_4 and v_5. Line $lv(q, v_2)$ has an intersection with $\mathrm{Bis}(q, p_4)$ in $\mathrm{ins}(v_{21})$ and $\mathrm{ins}(v_{22})$. Because one of the intersection is located between the point q and v_2, then this vertex is not considered as the region vertex; hence, the initial polygon will change. However, as an example, $lv(q, v_1)$ has an intersection with $\mathrm{Bis}(q, p_3)$ in $\mathrm{ins}(v_{11})$ and with $\mathrm{Bis}(q, p_4)$ in $\mathrm{ins}(v_{12})$. These intersection points are not located between q and vertex v_1; hence, vertex v_1 is considered as the region vertex. After examining the whole vertices, the new polygon region of RNN(q) will be Poly(v_1, v_3, v_4, v_5). □

Lemma 2
The set of intersection of bisectors will construct a polygon if each bisector has two region vertices.

Proof
The region is constructed from a set of region vertices, where each vertex is located in a bisector line. In a region, the bisector line is considered as the edge of the region. In a polygon, each edge has two vertices; hence, each bisector line that transforms into the edge of the polygon must have two vertices as well. If all bisectors have two vertices, then the set of the vertices will construct the region of RNN(q). □

Algorithms 1 and 2 are the common algorithms that will be used in the next method. Algorithm 1 describes about how to choose the region vertices from the list of vertices based on Lemma 1. Algorithm 2 shows how to determine if the set of vertices has created a polygon or is not based on Lemma 2.

```
Data: set of new Vertex V, last processed peer index ix
Result: Set of Region Vertex RV
$status \leftarrow true$;
$VT \leftarrow RV + V$; //assume V+RV are RV and stored in Vertex Temporary (VT)
$RV \leftarrow NULL$;
for $i \leftarrow 1$ to $sizeOf(VT)$ do
    $l \leftarrow line(VT[i], q)$; //create line l between vertex and q
    for $j \leftarrow 1$ to $ix$ do
        $int_p oint \leftarrow intersect(l, createBisector(q, pj))$;
        //get intersection between line l and bisector of pj
        if $dist(q, int_p oint) = dist(q, VT[i])$ then
            //ignore, vertex VT[i] is intersection point
        end
        else
            if $dist(q, VT[i]) = dist(int_p oint, VT[i]) + dist(int_p oint, q)$ then
                $status \leftarrow false$; //intersection point between VT[i] and q
            end
        end
    end
    if $(status = true)$ then
        $RV \leftarrow RV + VT[i]$;
    end
end
Return$RV$;
```

Algorithm 1: Region Vertex Verification

```
Data: set of Region Vertex RV
Result: True if Polygon, False if not Polygon
for $i \leftarrow 1$ to $sizeOf(RV)$ do
    $B \leftarrow getBisectors(RV[i])$;
    $CB \leftarrow addCount(B, 1)$; //add counter CB for each Bisectors according to B list
end
$polygon \leftarrow true$;
for $j \leftarrow 1$ to $sizeOf(CB)$ do
    if (getCount(CB[j])!=2 then
        $polygon \leftarrow false$;
    end
end
Return $polygon$;
```

Algorithm 2: Polygon Verification

4.1. Contact zone

Contact zone is a method to determine which next peers should be processed, and this method is based on the region vertex's influence to its surrounding. By using the contact zone, unnecessary peers that do not have bisector lines that will not cut though the initial region will be excluded.

Definition 6
The p_{last} is the last processed peer in region generation before the region of RNN(q) is finalize. The distance from q to this peer is defined as dist(q, p_{last}).

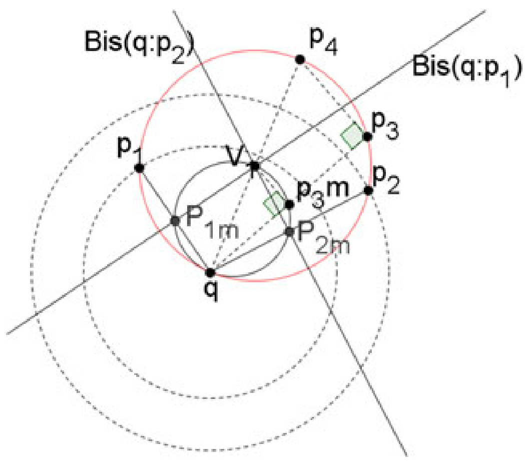


Figure 5. Possible peers location around vertex.

Because the peers are sorted by the distance from q and the processing sequence is started from the nearest to the farthest, there are no peers that can be processed if $\text{dist}(q, p_i) < \text{dist}(q, p_{\text{last}})$.

Lemma 3
The remaining peers P that can intersect with the region vertex RV are located in the region of Circ(rv_j,q) and $\text{dist}(q, p_i) \geqslant \text{dist}(q, rv_j)$

Proof
Let ω_1 and ω_2 be the two concentric circles with the center q. p_1 and p_2 are on ω_1,ω_2, respectively, and $\text{dist}(q, p_1) < \text{dist}(q, p_2)$. Let the perpendicular bisector of (q, p_1) and (q, p_2) intersect on v_1. Locus point p_3 is the point where the perpendicular bisector of (q, p_3) passes through v_1.

From Figure 5, let p_{1m}, p_{2m} be the midpoint of (q, p_1) and (q, p_2). Because $\angle(q, p_{1m}, v_1) = \angle(q, p_{2m}, v_1) = 90^o$, then we can conclude that (q, p_{1m}, v_1, p_{2m}) is a chord quadrangle. Let the circumcircle of (q, p_{1m}, v_1, p_{2m}) be ω_v with a diameter of (q, v_1).

Let p_{3m} be any point on ω_v. Because (q, v_1) is the diameter of ω_v, then $\angle(q, p_{3m}, v_1) = 90^o$. Let p_3 be a point on (q, p_{3m}) such that p_{3m} is the midpoint of (q, p_3); then, (p_{3m}, v_1) is the perpendicular bisector of (q, p_3). In this case, p_3 has a perpendicular bisector that passes through the polygon.

Let p_4 be a point on (q, v_1) such that V_1 is the midpoint of (q, p_4). Then, $\triangle(p_{3m}, q, v_1)$ is similar with $\triangle(p_3, q, p_4)$ with a ratio of 1:2. Because $\angle(q, p_{3m}, v1) = \angle(q, p_3, p4) = 90^o$, it can be concluded that the locus of p_3 is the circle with diameter (q, p_4). Let the circle be ω_4.

To prove that every point on circle ω_4 fulfilled the problem, let p_5 be any point on ω_4 and p_{5m} be the midpoint of (q, p_5). Then, it is obvious that $\triangle(p_4, p_5, q)$ similar with $\triangle(V_1, p_{5m}, q)$. Because (q, p_4) is the diameter of ω_4, then $\angle(p_4, p_5, q) = \angle(v_1, p_{5m}, q) = 90^o$. So, (v_1, p_{5m}) is the perpendicular bisector of (q, p_5).

It is shown that ω_4 is the only locus that is possible to intersect with the current vertex, and every point on ω_4 fulfilled the problem. The ω_4 is called the *contact zone*.

This proof shows that any peers located in the contact zone can have a perpendicular bisector that intersects the vertex where this vertex is the center of the contact zone. Therefore, any peers located inside this contact zone will have the perpendicular bisector that cuts through the bisectors that create the vertex. □

The contact zone is defined to specifically choose appropriate peers that really have an effect to the vertex, which means any points located near the vertex but do not affect the vertex will be excluded. This will bring us to another lemma that describes the area of the contact zone itself.

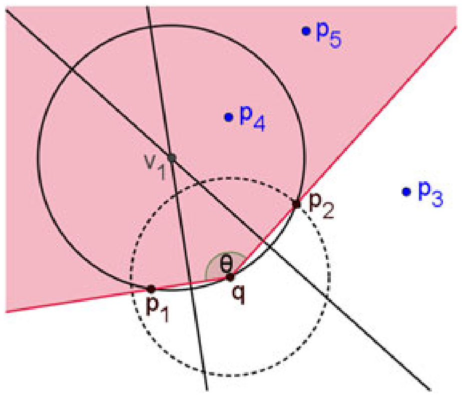


Figure 6. Pruning region of a vertex.

Lemma 4
Any peers will be considered in a contact zone if their distances are farther than p_{last} from q and shorter than from rv_j to q

$$\text{dist}(q, p_i) \geqslant \text{dist}(q, p_{last}) AND \text{dist}(rv_j, p_i) \leqslant \text{dist}(rv_j, q)$$

Proof
The proof in Lemma 3 shows that any peers located in the circle of contact zone will have a bisector line that intersects with the vertex. Hence, if the peers are located where $\text{dist}(v_1, p_i) \leqslant \text{dist}(v_1, q)$, the peers p_i will have a bisector line that cuts through the edge that constructs vertex v_1. □

Definition 7
Let v_1 be the vertex created from peers p_1 and p_2, and no initial region has been defined. A pruning region pr_y is the region that is created in $\angle(p_1, q, p_2)$

Lemma 5
If an initial region has not been defined, then all peers outside pruning region will be ignored, and contact zone is only applied to all peers in the pruning region.

Proof
If a non-initial region has not been created, the contact zone method can only exclude peers if they are outside the contact zone and inside the pruning region. Figure 6 shows the usage of contact zone in examining peers without the initial region. Peer p_3 will be ignored because it is located outside the pruning region, whereas p_5 will be excluded. Peer p_4 will be processed because it is located in contact zone. □

Lemma 6
Region of RNN(q) will always be part of the initial region of RNN(q).

Proof
Figure 4 shows the initial region of RNN(q) and the next processed peer p_4. The next processed peer will always cut through the polygon in certain vertex and reduce the area size of the initial region. Hence, the new region is part of the previous region, and the region of RNN(q) will always be less than or equal to the initial region of RNN(q). □

The purpose of contact zone is clear, to exclude unnecessary peers around the region vertex, which means for the next step, only peers that have a bisector line that cuts through the polygon in specified vertex will be processed. Because contact zone can only be applied in a vertex, there are two types of contact zone usage in determining the RNN region.

1. Contact zone with initial region: The contact zone is applied after the initial region is created. The number of contact zone will be equal to the number of vertices. If the initial region cannot be created, the sequential approach will be used, and whole peers will be processed.
2. Contact zone without initial region: The contact zone is immediately used after the first vertex has been created. This method effectively excludes unnecessary peers in he first round. This method will not process the whole peers, even if no polygon can be defined.

The next chapter will show how the contact zone is applied in generating the RNN region of a query point.

4.2. Contact zone with initial region

Contact zone can only be applied in a region vertex. The first method is to apply the contact zone after the initial region is created. Initial region is the first polygon created when processing the peers sequentially from the nearest to the farthest. The benefit of this approach is that it is easy to determine the next peers, because all peers have been sorted ascending on the basis of their distance to the query point q.

```
Data: Set of Sites S, query point q
Result: $R_1NN$ Region
$P \leftarrow sortAllPeer(q)$;//sort ascending all Sites S except q itself based on distance from q
while $(not(isEmpty(P))OR(not(isPolygon(V)))$ do
    $bi \leftarrow createBisector(q, pi)$; //get bisector between q and peer pi
    $Bis \leftarrow Bis + bi$;
    $V \leftarrow findIntersection(bi, Bis)$;
    $RV \leftarrow VerifyVertex(V, i)$;
    $lastPeer \leftarrow pi$;
    $getNext(pi)$;
end
if isPolygon(RV) then
    $findContactZone(RV)$; //get all contact zone around the polygon and exclude all
    peers outside the contact zone
    while not(isEmpty(P)) do
        $bi \leftarrow createBisector(q, pi)$;
        $Bis \leftarrow Bis + bi$;
        $V \leftarrow findIntersection(bi, Bis)$;
        $RV \leftarrow VerifyVertex(V, i)$;
        $lastPeer \leftarrow pi$;
        $findContactZone(RV)$;
    end
end
```

Algorithm 3: Contact Zone Pruning with Initial Region

The algorithm consists of two steps, where the second step depends on the result from the first step. The steps are as follows:

1. Construct initial region: The first step is to sort the peers based on the distance from the query point. Then, the peers are processed sequentially from the nearest until the initial region is created.
2. Apply contact zone: The second step is to apply the contact zone method on each vertex. After the contact zones have been applied, the next process would only process all peers inside the

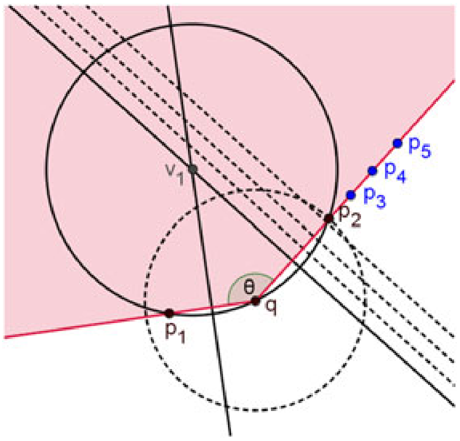


Figure 7. Parallel bisector.

contact zone and eliminate all peers outside the contact zone. This step is repeated until no more peers inside the contact zone exist.

From our experiments, contact zone can successfully eliminate all unnecessary peers in region generation; however, if the initial region cannot be created, the whole peers will be processed. This method produces good performance when initial region can be created with minimal use of peers; however, this method will produce a worst performance when the position of the peers is not allowed for the initial region to be created; for example, the peers are located in a straight line.

4.3. Contact zone without initial region

From Algorithm 3, the contact zone will not be used if the initial region cannot be created. To overcome this problem, contact zone will be used as soon as the first vertex is created. The main purpose of this method is to avoid processing the peers if they are located in a straight line. Figure 7 shows how contact zone with pruning region can avoid the execution of the peers where these peers are located in a straight line with currently executed peers.

This algorithm works in a similar way with Algorithm 3, except in examining the unused peers in contact zone; this method adds a parameter of pruning region, where this region is defined as the whole space in $\angle(p_1, q, p_2)$. The contact zone will only be applied to this pruning region, and any peers located inside this region and outside the contact zone will be eliminated. Our experiment shows that using this algorithm in the first place will reduce the number of peers needed in creating the RNN region. Hence, it is possible to generate RNN region in any query point position.

5. EVALUATION

In this section, the aim of the test is to analyze how efficient is the contact zone in reducing the unnecessary peers. We compare our result with another method that does not use the contact zone, but use the two distances of the farthest vertex as the boundary and processes the peers in the regions sequentially [18]. We will compare the efficiency of our method to point-to-point method in completing RNN query in the future. The data sets used are randomly generated from MATLAB, The MathWorks, Inc.3 Apple Hill DriveNatick, Massachusetts 01760 USA+1.508.647.7000 and spread uniformly on the space. The data vary from 10 up to 1000 peers distribution. The query point is chosen randomly on each data sets.

The first evaluation is comparing the total peers needed to create the RNN(q) region. Because we use two approaches that are with and without an initial region, we will first identify the peers needed to create the initial region. We use combined initial region with contact zone and (2*farthest vertex) approach [18]. However, the initial region in this method is using the sequential approach, where the peers are processed sequentially from the nearest distance until the polygon is created. Figure 8

```
**Data**: Set of Sites S, query point q
**Result**: $R_1NN$ Region
$P \leftarrow sortAllPeer(q)$;//sort ascending all Sites S except q itself based on distance from q
while $not(isEmpty(P))$ do
    $bi \leftarrow createBisector(q, pi)$;
    $Bis \leftarrow Bis + bi$;
    $pruni \leftarrow getPrunLine(q, pi)$; //get pruning line between q and pi
    if $i \geqslant 2$ then
        $V \leftarrow findIntersection(bi, Bis)$;
        $RV \leftarrow Verifyvertex(V, i)$;
        $PRc \leftarrow findPrunRegion(RV)$;//get pruning region in current vertex
        $lastPeer \leftarrow pi$;
        $CZ \leftarrow findContactZone(RV, PRn)$;//get all peers in Contact Zone in current vertex
        while not(isEmpty(CZ)) do
            $bj \leftarrow createBisector(q, CZ[j])$;
            $Bis \leftarrow Bis + bj$;
            $V \leftarrow findIntersection(bj, Bis)$;
            $RV \leftarrow VerifyVertex(V, i)$;
            $CZ \leftarrow findContactZone(RV, PRn)$;
        end
    end
    $getNext(pi)$;
end
```

Algorithm 4: Contact Zone Pruning without Initial Region

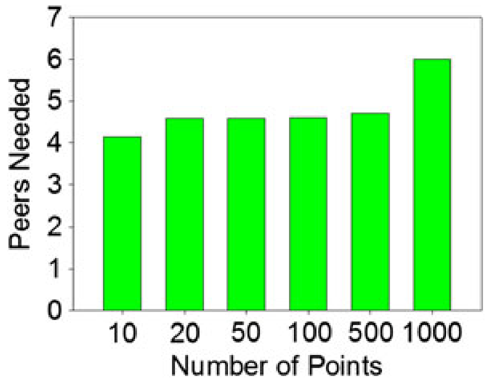
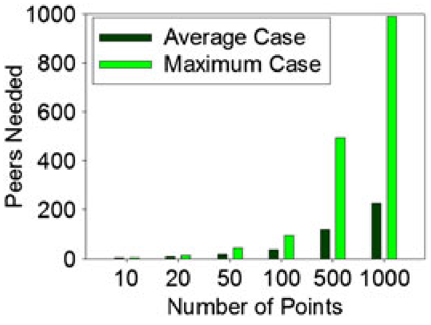


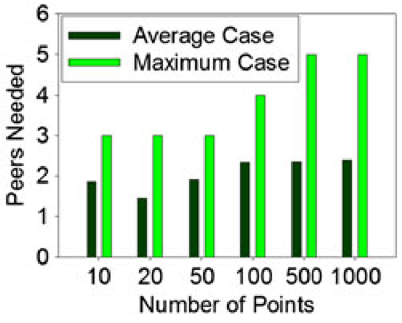
Figure 8. Peers needed for initial region.

shows the usage of peers in creating the initial region. The graph shows that on average, the number of peers needed in creating the initial region increases when the total number of points is increased. Because the complexity of the algorithm for the initial region is $O(n^3)$, the complexity of the algorithm increases when the number of peers increases. The next evaluation is analyzing the impact of contact zone in eliminating unnecessary peers after the initial region is created. The number of peers processed after the initial region is created will show the effectiveness of the elimination process. The less number of peers processed, the higher number of peers eliminated. However, if the initial region cannot be created, then the algorithm will have to process all peers in the space.

Figure 9 shows the number of peers needed to create the RNN region after the initial region is created. The number of peers needed increased significantly when the number of points in space increased by using the 2*farthest vertex limit; meanwhile, the number of peers needed by using the contact zone remains stable across multiple number of points.

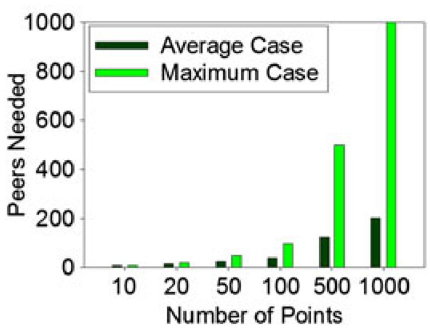


(a) 2*Farthest vertex

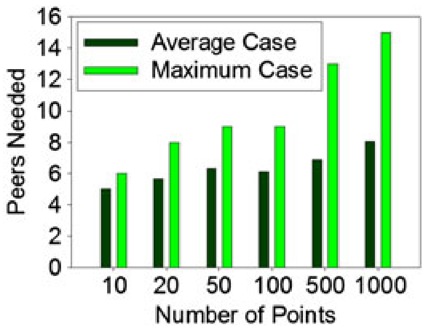


(b) Contact Zone

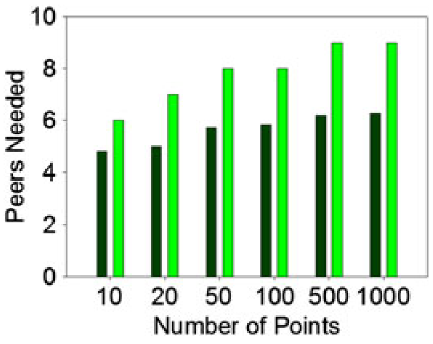
Figure 9. Peers needed for reverse nearest neighbor region after initial region. (a) 2*Farthest vertex and (b) contact zone.



(a) 2*Farthest vertex



(b) Initial with CZ



(c) Contact Zone

Figure 10. Peers needed for reverse nearest neighbor region. (a) 2*Farthest vertex, (b) initial with contact zone (CZ), and (c) contact zone.

Figure 10 shows the total peers needed to be processed for creating RNN region from a specific query point. With the (a) farthest vertex approach, both the average and worst cases will process more peers significantly as the number of peers increased, whereas on the worst case, the number of peers needed to be processed is almost all peers in the space. Meanwhile, by using the (b) contact zone after the initial region is created, the total peers needed to be processed only show slight increment in both the average and worst cases. The same pattern is also shown in region creation with (c) contact zone without initial region. However, the number of peers needed for Contact Zone without initial region is smaller from Contact Zone with initial region, since some peers are needed to create the initial region. These results show that contact zone can effectively choose the best peers to create the RNN region. However, if contact zone is used with initial region and the initial zone cannot be created, the contact zone will not be processed. According to this situation, it is best that contact zone is used without initial region.

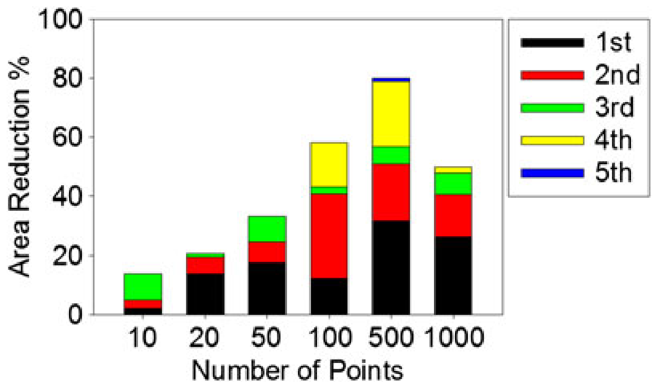


Figure 11. Area reduction with contact zone.

To analyze the effect of Contact Zone in reducing the area size of RNN region and initial region, we observe area reduction on each peer processed. As shown in the figure 11, on the average, the size of area reduction increase when the number of peers are increased. This means the size of RNN region is more smaller than initial region, hence initial region cannot represent the RNN region.

6. CONCLUSIONS

In this paper, we introduce the concept of contact zone, where we emphasize the efficiency of algorithm in region pruning in determining the region from any generator points. This method can be used with and without an initial region. We presented a detailed theoretical analysis and algorithm, and also different study cases. Our experiments show that the contact zone without initial region can determine the region of RNN from any query point with smaller number of peers than any method with initial region. Our experiment also shows that contact zone can easily identify the points that will be involved in RNN region generation; hence, this method will be suitable in constructing RNN region in mobile devices, where there is no need to process whole points in the space to obtain the region of RNN(q). Although this method can identify the RNN region for the query point without having to know the objects position because any objects located in the region will be considered as the RNN(q), our future works will be focused on efficiency comparison between the RNN by region and point-to-point approaches in solving the RNN query.

ACKNOWLEDGEMENT

The first author's research is funded by the Directorate General of Higher Education of Indonesia (batch 5-2011).

REFERENCES

1. Waluyo AB, Srinivasan B, Taniar D. Research in mobile database query optimization and processing. *Mobile Information Systems* 2005; **1**(4):225–252.
2. Xuan K, Zhao G, Taniar D, Srinivasan B. Continuous range search query processing in mobile navigation. In *ICPADS*. IEEE: Melbourne, Victoria, Australia, 2008; 361–368.
3. AL-Khalidi H, Taniar D, Safar M. Approximate algorithms for static and continuous range queries in mobile navigation. *Computing* Springer Vienna, 2012:1–28. DOI: 10.1007/s00607-012-0219-7.
4. Waluyo AB, Taniar D, Rahayu JW, Srinivasan B. Mobile service oriented architectures for NN-queries. *Journal of Network and Computer Applications* March 2009; **32**(2):434–447.
5. Zhao G, Xuan K, Taniar D. Path kNN query processing in mobile systems. *IEEE Transactions on Industrial Electronics* 2013; **60**(3):1099–1107.
6. Xuan K, Zhao G, Taniar D, Safar M, Srinivasan B. Voronoi-based multi-level range search in mobile navigation. *Multimedia Tools and Applications* June 2011; **53**(2):pp 459–479, Springer US.
7. Cho H-J, Choe S-K, Chung T-S. A distributed approach to continuous monitoring of constrained k-nearest neighbor queries in road networks. *Mobile Information Systems* 2012; **8**(2):107–126.

8. Cho H-J, Kwon SJ, Chung T-S. A safe exit algorithm for continuous nearest neighbor monitoring in road networks. *Mobile Information Systems* 2013; **9**(1):37–53.
9. Tran Q, Taniar D, Safar M. Reverse k nearest neighbor and reverse farthest neighbor search on spatial networks. In *Transactions on Large-Scale Data- and Knowledge-Centered Systems I*, Vol. 5740, Lecture Notes in Computer Science. Springer Berlin: Heidelberg, 2009; 353–372.
10. Taniar D, Safar M, Tran QT, Rahayu JW, Park JH. Spatial network RNN queries in GIS. *The Computer Journal* 2011; **54**(4):617–627.
11. Korn F, Muthukrishnan S. SIGMOD '00. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ACM New York, NY, USA, 2000; 201–212.
12. Benetis R, Jensen S, Karciauskas G, Saltenis S. Nearest and reverse nearest neighbor queries for moving objects. *The VLDB Journal* 2006; **15**(3):229–249.
13. Tao Y, Papadias D, Lian X. Reverse kNN search in arbitrary dimensionality. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04. VLDB Endowment: Toronto, Canada, 2004; 744–755.
14. Singh A, Ferhatosmanoglu H, Tosun Ac. High dimensional reverse nearest neighbor queries. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03. ACM: New York, NY, USA, 2003; 91–98.
15. Achtert E, Kriegel H-P, Kröger P, Renz M, Züfle A. Reverse k-nearest neighbor search in dynamic and general metric databases. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09. ACM: New York, NY, USA, 2009; 886–897.
16. Hu L, Ku W-S, Bakiras S, Shahabi C. Verifying spatial queries using voronoi neighbors. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10. ACM: New York, NY, USA, 2010; 350–359.
17. Wu W, Yang F, Chan C-Y, Tan K-L. FINCH: evaluating reverse k-nearest-neighbor queries on location data. *Proceedings of the VLDB Endowment*, Volume 1, Issue 1, August 2008; 1056–1067.
18. Cheema M, Lin X, Zhang W, Zhang Y. Influence zone: efficiently processing reverse k nearest neighbors queries. *In 2011 IEEE 27th International Conference on Data Engineering (ICDE)*, Hannover, Germany, April 2011; 577–588.
19. Stanoi I, Riedewald M, Agrawal D, Abbadi AE. Discovery of influence sets in frequently updated databases. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01. Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2001; 99–108.
20. Okabe A, Boots B, Sugihara K, Chiu SN. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, Wiley Series in Probability and Statistics. Wiley: Baffins Lane, Chichester, West Sussex, England, 2009.
21. Safar M, Ebrahimi D, Taniar D. Voronoi-based reverse nearest neighbor query processing on spatial networks. *Multimedia Systems* 2009; **15**(5):295–308.
22. Zhao G, Xuan K, Rahayu W, Taniar D, Safar M, Gavrilova ML, Srinivasan B. Voronoi-based continuous k nearest neighbor search in mobile navigation. *IEEE Transactions on Industrial Electronics* 2011; **58**(6):2247–2257.
23. Tran QT, Taniar D, Safar M. Bichromatic reverse nearest-neighbor search in mobile systems. *Systems Journal, IEEE* 2010; **4**(2):230–242.
24. Yang C, Lin K-I. An index structure for efficient reverse nearest neighbor queries. *In 17th International Conference on Data Engineering, 2001. Proceedings*, Heidelberg, Germany, 2001; 485–492.
25. Lu J, Lu Y, Cong G. Reverse spatial and textual k nearest neighbor search. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11. ACM: New York, NY, USA, 2011; 349–360.
26. Achtert E, Böhm C, Kröger P, Kunath P, Pryakhin A, Renz M. Efficient reverse k-nearest neighbor search in arbitrary metric spaces. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06. ACM: New York, NY, USA, 2006; 515–526.
27. Taniar D, Rahayu JW. A taxonomy of indexing schemes for parallel database systems. *Distributed and Parallel Databases* 2002; **12**(1):73–106.
28. Taniar D, Rahayu JW. Global parallel index for multi-processors database systems. *Information Sciences* 2004; **165**(1–2):103–127.
29. Alamri S, Taniar D, Safar M. Indexing moving objects for directions and velocities queries. *Information Systems Frontiers* Springer US, 2013; **15**:235–248. DOI: 10.1007/s10796-012-9367-8.
30. Kolahdouzan M, Shahabi C. Voronoi-based k nearest neighbor search for spatial network databases. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04. VLDB Endowment: Toronto, Canada, 2004; 840–851.
31. Sharifzadeh M, Shahabi C. Approximate Voronoi cell computation on spatial data streams. *The VLDB Journal* 2009; **18**(1):57–75.
32. Sharifzadeh M, Shahabi C. Vor-tree: R-trees with voronoi diagrams for efficient processing of spatial nearest neighbor queries. *Proceedings of the VLDB Endowment*, Volume 3, Issue 1; 1231–1242.
33. Xuan K, Zhao G, Taniar D, Rahayu W, Safar M, Srinivasan B. Voronoi-based range and continuous range query processing in mobile databases. *Journal of Computer and System Sciences* 2011; **77**(4):637–651. JCSS IEEE AINA 2009.
34. Bohan L, Xiaolin Q. Research on reverse nearest neighbor queries using ranked voronoi diagram. *In 2009 1st International Conference on Information Science and Engineering (ICISE)*, Nanjing, China, December 2009; 951–955.