Roaming Regions for Delaunay Nodes

Laxmi P Gewali and Romas Hada
University of Nevada, Las Vegas
laxmi.gewali@unlv.edu

Abstract

Delaunay graphs have been used in CAD/CAM, sensor networks, and geographic information system. We investigate the reliability properties of nodes in Delaunay graphs. For measuring the reliability we formulate the concept of roaming-region for nodes. The *roaming-region $R(i)$* of a Delaunay node v_i is such that the Delaunay graph does not change as long as v_i remains within $R(i)$. A node v_i with large roaming region $R(i)$ such that v_i is positioned near the center of $R(i)$ is identified as a reliable node. Two types of roaming regions called *(i) lateral roaming region $LR(i)$* and *(ii) radial roaming region $RR(i)$* are distinguished to develop the algorithm. The roaming region itself is expressed as the intersection of $RR(i)$ and $LR(i)$. For nodes inside the convex hull, called *deep internal nodes*, we present an $O(n^2)$ time algorithm for computing their roaming region, where n is the number of nodes in the Delaunay triangulation. We finally discuss generalization and extension of the proposed algorithm.

1 Introduction

A network or graph consists of a set of nodes V and a set of edges E. An edge $e \in E$ connects two nodes in V. Such a network is usually denoted by $G(V, E)$. The term vertex is also used to indicate node. Similarly, the term link is also used to indicate edge. A class of simple networks used extensively in sensor networks and geographical information system is the *planar network*. It is noted that a network is called *planar* if it can be drawn in the plane without intersecting edges. *Delaunay triangulation, relative neighborhood graph, Gabrial graph* are examples of widely used planar network. One of the main reasons for the popularity of planar graphs in application areas is the fact that the size of any planar graph (number of edges) is not large. In fact, in a planar graph the number of vertices and the number of edges are linearly related. Furthermore, the data structure for representing planar graphs are much simpler and can be updated quickly.

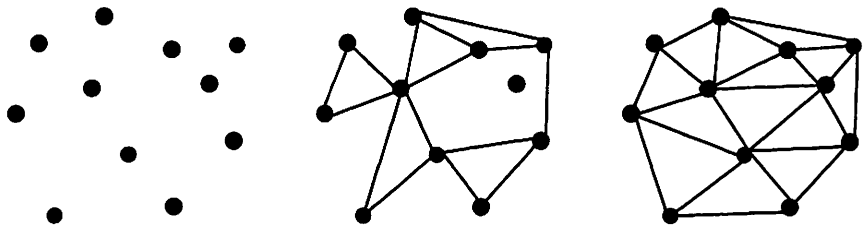

Figure 1: Illustrating a set of nodes (left), partial triangulation (center) and maximal triangulation (right)

In this paper we consider the reliability properties of planar network when nodes of the network are allowed to change slightly in their neighborhood. Broadly speaking, a node in a network is called *reliable* if the connectivity of the network does not change when the node moves slightly from the initial position. In particular, we investigate the reliability properties of nodes in a Delaunay triangulation. In Section 2, we review properties and algorithms for Delaunay triangulation and related structures.

In Section 3, we present the main contribution of the paper. We first formulate the notion of *roaming-region* for a node of Delaunay triangulation. We show that as long as a node remains within its roaming-region, the underlying Delaunay network does not change. We then present an efficient algorithms for computing the roaming-region for an internal node in the Delaunay triangulation. In Section 4, we discuss the extensions of the proposed algorithm and its application for measuring the reliability of Delaunay nodes.

2 Preliminaries

In this section we present a brief overview of algorithms for triangulation and node relocation in reference to sensor network applications.

2.1 Triangulated Network

Triangulation of a set of point sites $S = \{p_0, p_1, p_2, \ldots, p_{n-1}\}$ is the construction of the maximum number of non-intersecting triangles with vertices in S. Triangulations could be partial or maximal as illustrated in Figure 1.

The term *triangulation* is generally understood to mean maximal triangulation. The triangulation problem has been investigated and used in many branches of science and engineering that include surveying, cartography, robotics, and geographic information system [6].

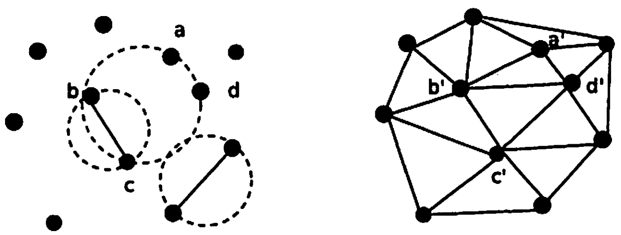

Figure 2: Delaunay edges shown in left and corresponding Delaunay triangulation of given point sites in right

After the advent of computational geometry in mid 1970's, there was a flurry of research activities dealing with the development of efficient algorithms for triangulation [9]. It is noted that a given set of point sites can be triangulated in exponentially many ways [9].

A triangulation with many interesting properties is the *Delaunay triangulation.* A Delaunay triangulation is the dual of Voronoi diagram [6, 9]. An interesting property of Delaunay triangulation is the fact that it maximizes the smallest angle of the triangulation [6]. There is a direct characterization of Delaunay triangulation in term of *in-circle property* i.e. two points b and c are the end points of a Delaunay edge if and only if there is a circle through b and c that passes through no other point sites and contains no sites in its interior. Consequently, all triangles in the Delaunay triangulation for a given point sites will have empty circumscribed circles. It is always unique as long as no four points in the given point sites are co-circular. If more than three point-sites are co-circular then there will not be unique Delaunay triangles.

In Figure 2, a circle through point sites b and c contains no other point site inside it. Hence, it is a Delaunay edge for given point sites. Similarly for point sites a, b, c and d as shown in the left of Figure 2, the segment connecting point sites c and a cannot be a Delaunay edge as the circle passing through point sites c, d and a is not an empty circle as it encloses a point site b.

As Delaunay triangulation maximizes smallest angle, it is geometrically nice and, in general, pleasing to the eye. Delaunay triangulations have a number of interesting properties other than the empty circle property [9, 6].

Delaunay triangulation can be computed by using divide and conquer algorithm or incremental algorithm [6]. The most popular algorithm for computing Delaunay triangulation is based on plane sweep and runs in $O(nlogn)$ time [6].

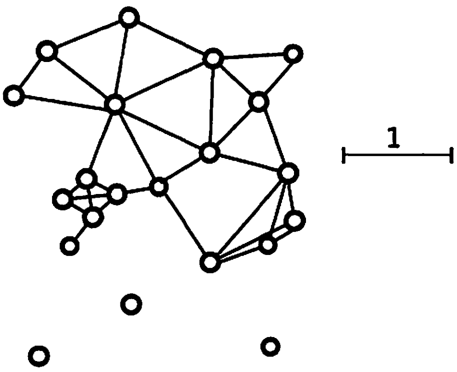

Figure 3: Illustrating unit disk graph

2.2 Algorithm for Node Relocation

An interesting problem in network design is the reconstruction of a network when nodes are allowed to change their position. Not much research work has been reported on the change of network when node position varies. Some recent work on node relocation have been considered by investigators in sensor network community. For example, Coskum [4] has investigated connected cover problem for sensor network when some sensor nodes are allowed to change location. Rongratana et. al. [7, 8] have addressed the problem of identifying *free-regions* of a sensor node so that the connectivity of network is preserved as long as a node remains within its free-region. Since the contribution presented in Section 3 is also dealing with the relocation of nodes we describe an overview of the concepts and results reported in [7, 8]; where the relocation is done for nodes of unit disk graph.

Unit disk graph is a very useful concept for application in sensor network. Unit disk graph is defined when all nodes have identical transmission range [3] which is taken without loss of generality as 1. Each sensor node becomes the vertex of the unit disk graph. Two nodes v_1 and v_2 are connected by an edge if the distance between v_1 and v_2 is less than or equal to 1. Basically, a unit disk graph (UDG) is obtained by connecting all vertices that are within the transmission range (=1).

Figure 3 shows an example of UDG with indicated range 1. It is remarked that UDG becomes very dense if the transmission range is large. Let $TD(i)$ denote the transmission disk of node v_i.

The free-region of a node is computed in terms of *in-free region* and *out-free region* of that node. Consider a sensor node v_1 whose neighbor nodes in the unit disk graph are v_2, v_3 and v_4 as shown in Figure 4. A node v_j is called *out-bound node* of node v_i if (i) v_j lies outside the transmission

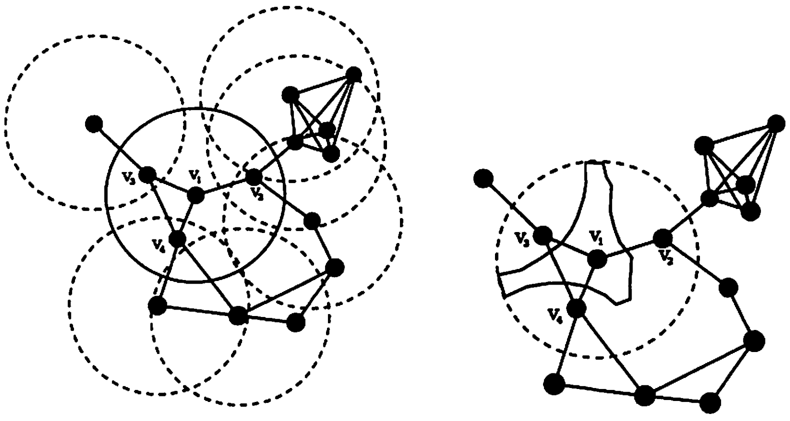

Figure 4: Illustrating an out-free-region of a node

disk $TD(i)$ of v_i and (ii) the transmission disks $TD(i)$ and $TD(j)$ intersect. The disks of out-bound nodes of v_1 are drawn dashed in the left side of Figure 4.

The region of $TD(i)$ that is not intersecting with the transmission disks of outbound nodes of v_1 gives the *out-free region* of v_1 as shown in right side of Figure 4 . Similarly, the notion of *in-free region* is considered. *In-bound* nodes of v_1 are the nodes that lies within its transmission disk. The intersection of the transmission disk of *in-bound nodes* give the in-free region as shown in Figure 5. The detail are reported in [7, 8].

The intersection of *in-free region* and *out-free region* precisely gives the *free region* of node v_1. As long as node v_1 remains within its free-region the unit disk graph doesn't change. Free-region of a sensor node can be computed in $O(k^2)$ time, where k is the number of out-bound and in-bound nodes of v_1 [7, 8]. It is interesting to note that the problem of computing free region of a sensor node has lower bound $\Omega(klogk)$ which is proved in [7, 8] by reducing the sorting problem to free-region computation problem.

3 Roaming Region for Delaunay Nodes

3.1 Problem Formulation

Consider a set S of nine nodes as shown in the left side of Figure 6. The Delaunay triangulation of these nodes is shown in the right side.

If we move a node slightly then it is very likely that the Delaunay triangulation will not change. On the other hand if we continue to move a node significantly further from its initial position then the resulting Delau-

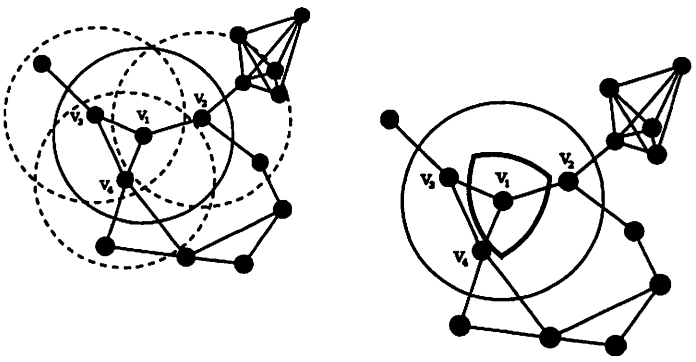

Figure 5: Illustrating an in-free-region of node v_1 (bounded by three circular arcs)

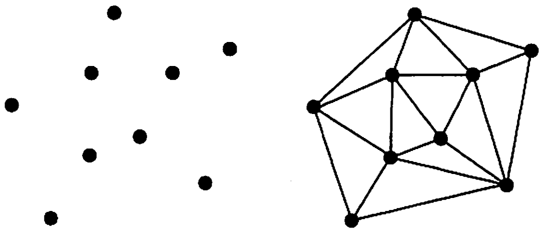
Figure 6: Illustrating Delaunay triangulation

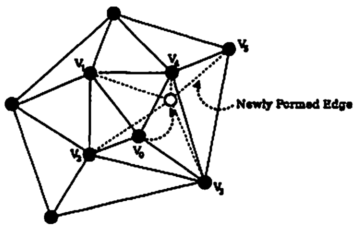

Figure 7: Illustrating the change of triangulation by node movement

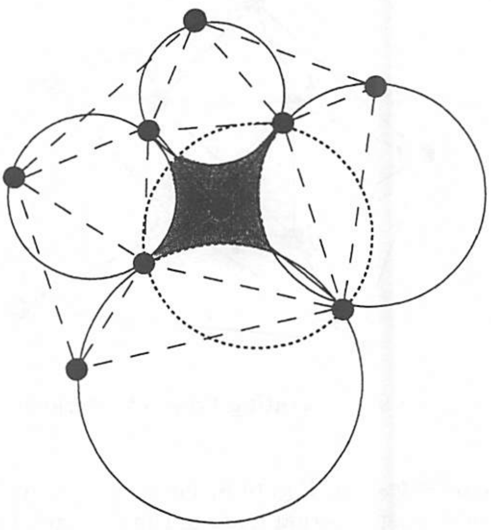

Figure 8: Illustrating roaming-region (shaded)

nay triangulation changes. This change of Delaunay triangulation is shown in Figure 7. In Figure 7, initially node v_0 is connected directly to nodes v_1, v_2, v_3 and v_4. When v_0 is moved to new position as indicated in the figure, it will be connected to one more node which is v_5.

This observation shows that it would be interesting to determine the connected region for a node such that the Delaunay triangulation remains same no matter where the node is placed in the region. To formulate this problem formally we extend the free-region concept introduced in [7] for unit disk graph.

Definition 1 (*Roaming Region*): Consider the Delaunay triangulation of a set $S = \{v_0, v_1, v_2, ..., v_{n-1}\}$ of points in the plane. The *roaming-region* of any node v_i is the maximal region R_m(i) in the proximity of v_i such that the Delaunay triangulation does not change when v_i is moved to any point within $R_m(i)$. An example of roaming region is shown in Figure 8. The shaded area is the roaming region.

To develop an algortihm for constructing roaming reason for nodes it is necessary to distinguish nodes into several classes. Nodes that are on the boundary of the convex hull of nodes are called *external nodes* and those that are inside the convex hull are the *internal nodes.* The internal nodes can be further distinguished as *shallow-internal* nodes and *deep-internal* nodes.

Defintion 2 (*Deep Internal*): Consider the Delaunay Triangulation T of given nodes. An internal node v_i is called *deep-internal* if all neighbors

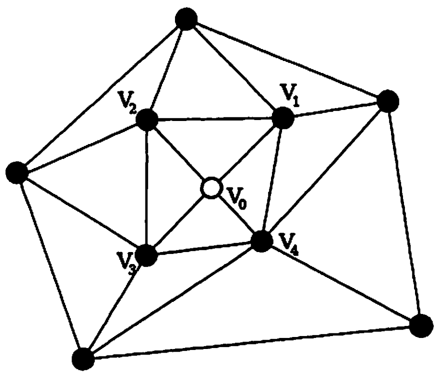

Figure 9: Illustrating types of interior node

of v_i are internal nodes. In Figure 9, node v_0 is a *deep-internal.* In fact node v_0 is the only deep-internal node in the triangulation of Figure 9.

Definition 3 (*Shallow Internal*): An internal node v_i is called *shallow-internal* if some neighbor of v_i is an external node. In Figure 9, four nodes v_1, v_2, v_3, and v_4 are shallow-internal nodes.

Remark 1: It is remarked that the set of nodes in a Delaunay triangulation T can be viewed as the union of three disjoint sets: (i) external nodes, (ii) deep-internal nodes and (iii) shallow-internal nodes.

3.2 Roaming Region for Deep-Internal Nodes

Before developing methods for computing roaming region, we recall one of the key properties of Delaunay triangulation which states that the circum circle of any triangle of a Delaunay triangulation is empty i.e. the circum circle does not contain any other node of the triangulation. This property is called "empty-circle property". Our method of computing roaming region of a node is based on the use of set intersection and set differences of circum circles of carefully selected triangles (both Delaunay and non-Delaunay) in the proximity of the candidate node. For this purpose, we start with the characterization of *radial* and *lateral* triangles for a given node v_i as follows.

Definition 4 (*Radial Triangle*): Consider a candidate Delaunay node v_i. Let $t_1, t_2, t_3, ...t_k$ be the triangles incident on v_i. If the degree of v_i is k then there are k incident triangles. The triangles sharing the sides of incident triangles opposite to v_i are the *radial triangles* of v_i. We use notations $r_1, r_2, r_3, ..., r_m$ to denote radial triangles. In Figure 10, radial triangles for node v_i are shown shaded grey.

An inspection of a Delaunay triangulation diagram reveals that only some nodes have radial triangles corresponding each of its incident triangle.

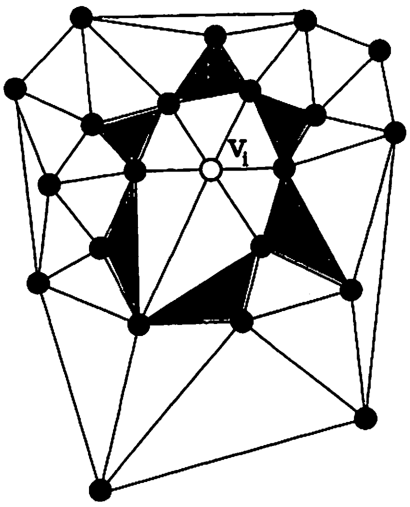

Figure 10: Illustrating radial triangles for node v_i

This leads to the following remark.

Remark 2: For a deep-internal node v_i there will be radial triangles for each incident triangle of v_i.

The notion of (*Lateral Triangles*) are captured by considering consecutive incident triangles t_i and t_{i+1} for a candidate node v_i.

Definition 5 *Lateral Triangles*: Let $t_1, t_2, ...t_k$ be the triangles incident on node v_i such that they are ordered angularly around v_i. A pair of consecutive incident triangles t_i and t_{i+1} form a quadrilateral $v_i v_p v_q v_r$ incident at v_i. The non-Delaunay triangle $v_p v_q v_r$ is a *lateral triangle* for vertex v_i. In Figure 11, only two out of six possible lateral triangles are shown.

Formation of Radial Roaming Region $RR(i)$: Let $R_1, R_2, R_3, ...R_n$ be radial disks of v_0. Here notation R_i is used to denote the radial disk enclosed by circum circle of radial triangle r_i. Let $v_1, v_2, v_3, ...v_m$ be the neighbor nodes of v_i. We use D_{max} to denote the convex hull region of the point sites including v_i and neighbor nodes $v_1, v_2, v_3, ...v_m$. Then, radial roaming region $RR(i)$ of v_i is obtained by removing regions occupied by radial disks from D_{max} which can be expressed as follows.

$$\mathrm{RR(i)} = D_{max} - R_1 - R_2 ..... - R_n \;...\; \text{(i)}$$

In Figure 12, four radial disks R_1, R_2, R_3 and R_4 are formed corresponding to Delaunay triangles D_1, D_2, D_3 and D_4 respectively. The radial roaming region $RR(0)$ for node v_0 is shown on the right side which is enclosed by four circular arcs.

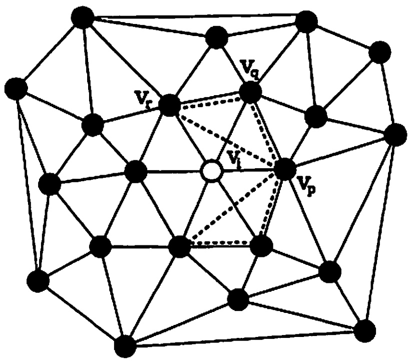

Figure 11: Illustrating lateral triangles for node v_i (only two are shown)

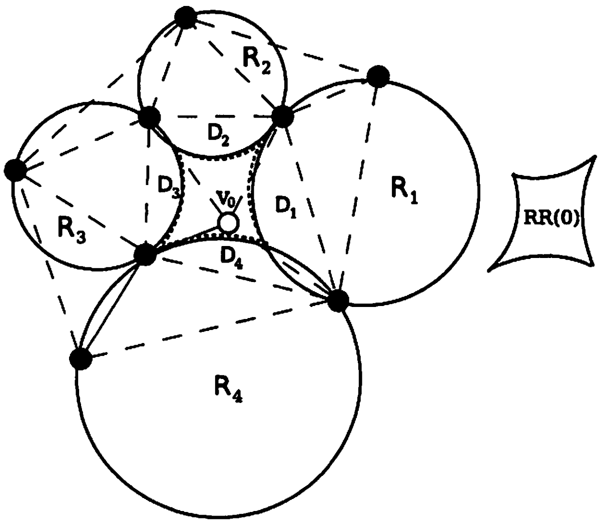

Figure 12: Illustrating radial roaming region of v_0

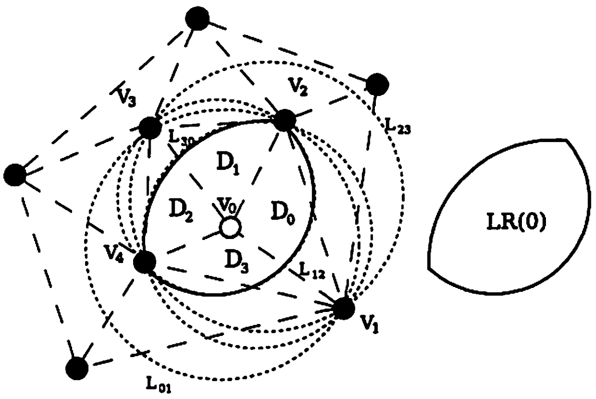

Figure 13: Illustrating lateral circum circles for v_0

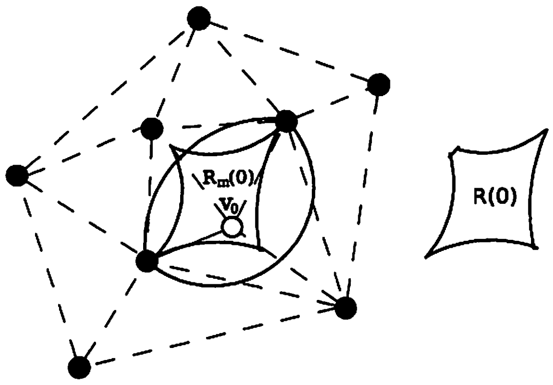

Figure 14: Illustrating formation of roaming region R(0)

Observation 1: Let $N_i = \{v_{i_1}, v_{i_2}, v_{i_3}, ..., v_{i_m}\}$ be the neighbor nodes of v_i. As long as as v_i remains within $RR(i)$, no new neighbors are formed for v_i. However, some existing neighbors may cease to be neighbor.

Formation of Lateral Roaming Region $LR(i)$: The lateral roaming region of a node is formed by the overlay of disks corresponding to the circum circles of lateral triangles. Let $L_1, L_2, L_3, ..., L_k$ be the lateral disks for node v_i. Then the *lateral roaming region LR(i)* for node v_i is given by the intersection of lateral disks as expressed below.

$$LR(i) = \cap(L_1, L_2, L_3, ...L_k) \text{ ... (ii)}$$

Observation 2: As long as a node v_i remain within $LR(i)$, existing initial neighbors will continue to be neighbors. However, additional neighbors may be formed.

The formation of lateral roaming region is illustrated in Figure 13, where lateral disks are drawn by dashed circles. The intersection of these disks is the lateral roaming region $LR(0)$ for node v_0, shown on the right side of Figure 13, which is enclosed by two circular arcs.

Observation 1 and Observation 2 imply that the roaming region $R(i)$ for node v_i is given by the intersection of lateral roaming region $LR(i)$ and radial roaming region $RR(i)$ as expressed below.

$$R(i) = \cap(LR(i), RR(i)) \text{ ... (iii)}$$

The overlay of lateral roaming region $LR(0)$ and radial roaming region $RR(0)$ for node v_0 is shown in Figure 14. The intersection of $LR(0)$ and $RR(0)$ which is the roaming region $R(0)$ of v_0 is shown on the right side of the figure.

The algorithm for computing roaming region for internal nodes is given below.

```
Algorithm 1: Roaming Region for Internal Nodes

  Input:  (i) Delaunay Triangulation $DT$
          (ii) Interior node $v_i$
  Output: Roaming region R(i) for $v_i$

  Step 1: a. Determine all neighbor nodes $v_1, v_2, v_3, ..., v_m$ in $DT$
             b. Let $D_{max}$ be the convex hull region of neighbors of $v_i$
  Step 2: i.  Determine radial triangles $t_1, t_2, t_3, ..., t_k$ for node $v_t$.
             ii. Computer radial disks $R_1, R_2, ..., R_k$ corresponding to
                 $t_1, t_2, t_3, ..., t_k$
  Step 3:    Determine $RR(i) = D_{max} - R_1 - R_2 - R_3 ... - R_k$
  Step 4:    Identify lateral disks $L_1, L_2, L_3, ...L_p$ for vertex $v_i$
```

Step 5: Compute $LR(i) = \cap(L_1, L_2, L_3, ..., L_p)$
Step 6: Determine the output $R(i) = \cup(LR(i), RR(i))$

Theorem 1: The roaming region of a deep internal node v_i in a Delaunay triangulation DT can be computed in $O(n^2)$ time.

Proof: We assume that the Delaunay triangulation is available in doubly connected edge list data structure. Step 1 can be done in $O(d(v_i))$ time by following the edge list incident on v_i, where $d(v_i)$ is the degree of node v_i. Since the number of edges in DT are linearly related to number of vertices n, $d(v_i) = n$. Hence Step 1 and Step 2 take $O(n)$ time. Once radial triangles around v_i are available, radial disks can be determined in $O(n)$ time. By navigating doubly connected edge list data structure, lateral triangles of v_i can be determined in $O(n)$ time. Finally, the intersection of lateral disks (Step 5) and intersection of $LR(i)$ and $RR(i)$ can be done in $O(n^2)$ time in straightforward manner. Thus the entire algorithm takes $O(n^2)$ time. □

4 Discussion

We formulated the notion of roaming region for nodes in a Delaunay triangulation. As long as a node remains within its roaming region the underlying Delaunay triangulation does not change. To capture the roaming region of a node in Delaunay triangulation we characterize two kinds of triangles in its proximity: radial triangles and lateral triangles. Radial triangles are used to compute radial roaming region and lateral triangles are used to compute lateral roaming region. The roaming region $R(i)$ of node v_i is expressed as the intersection lateral roaming region $LR(i)$ and radial roaming region $RR(i)$.

The concept of roaming region can be applied for measuring reliability of nodes in sensor networks. A sensor node with large roaming region such that the node is near the center of its roaming region is reliable in the sense that the connectivity does not change for small movement of the node.

The algorithm for computing roaming region presented in this paper runs in $O(n^2)$ time. By seeking more insight into the structure of roaming region it may be possible to develop faster algorithm. The algorithm works only for deep-internal node. It should be straightforward to generalize it to shallow-internal nodes. We made some preliminary examination of the roaming region for external nodes i.e., nodes on the boundary of the convex hull. It turns out that the roaming region of external nodes are not necessarily bounded. Based on this observation it is interesting to consider roaming regions for boundary nodes such that the regions lie inside the convex hull of input nodes.

References

[1] K. Q. Brown. Voronoi diagrams from convex hulls. *Information Processing Letters*, 9(5):223–228, 1979.

[2] L. Paul Chew. Constrained delaunay triangulation. *Algorithmica*, 4:97-108, 1989.

[3] B. N. Clark, C. J. Colbourn, D. S. Johnson, Unit Disks Graphs, *Discrete Mathematics*, 86:165-177, 1990.

[4] Vedat Coskum. Relocating sensor nodes to maximize cumulative connected coverage in wireless sensor networks. *Sensors*, 8, 2008.

[5] B. N. Delaunay. Sur la sphre vide: Izv. akad. nauk sssr. *RNauk SSSR, Otdel. Mat. Est. Nauk*, 7:793–800, 1934.

[6] Marc van Kreveld Mark Overmars Mark de Berg, Otfried Cheong. *Computational Geometry: Algorit3hms and Applications.* Springer-Verlag, 2008.

[7] Jan B. Pedersen N. Rongratana, Laxmi Gewali. Estimating the free region of a sensor node. *JCMCC*, 74:53–64, 2010.

[8] Henry Selvaraj Navin Rongratana, Laxmi Gewali. Characterizing free-regions of sensor nodes. *International Conference on Systems Engineering*, pages 375–379, 2008.

[9] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, 1998.

[10] Jim Ruppert. A delaunay refinement algorithm for quality 2-dimensional mesh generation. *NASA Ames Research Center*, 1993.