2009

Free Regions of Sensor Nodes

Laxmi P. Gewali
University of Nevada, Las Vegas, laxmi@cs.unlv.edu

Navin Rongatana
HRH College of Engineering, nrongra@gmail.com

Henry Selvaraj
University of Nevada, Las Vegas, henry.selvaraj@unlv.edu

Jan B. Pedersen
University of Nevada, Las Vegas

Systems Science, Vol. 35, No. 2, 2009

LAXMI GEWALI*, NAVIN RONGATANA*,
HENRY SELVARAJ*, JAN B. PEDERSEN*

FREE REGIONS OF SENSOR NODES

We introduce the notion of **free region** of a node in a sensor network. Intuitively, a **free region** of a node is the connected set of points R in its neighborhood such that the connectivity of the network remains the same when the node is moved to any point in R. We characterize several properties of free regions and develop an efficient algorithm for computing them. We capture free region in terms of related notions called **in-free region** and **out-free region**. We present an $O(n^2)$ algorithm for constructing the free region of a node, where n is the number of nodes in the network.

Keywords: *sensor network, node relocation, free region*

1. Introduction

Consider n sensor nodes v_1, v_2, ..., v_n deployed on a terrain surface, which is taken as a two dimensional plane. The location of node v_i is represented by point q_i with coordinates x_i and y_i, respectively. The transmission range r of all sensor nodes is assumed to be identical and the implied transmission region is taken as the **transmission disk** *TD*(i) of radius r. The circle of the transmission (i.e., perimeter) is denoted as *TC*(i). We can imagine a network obtained by connecting all pairs of nodes within each others' transmission range. Such a network is often called **Unit Disk Graph (UDG)** [1, 12] and we denote it by $G(V, E)$, where V and E are the set of nodes and the set of edges, respectively. Figure 1 shows an example of the unit disk graph induced by 14 nodes, where the disk with dashed boundary indicates the transmission region corresponding to node v_1.

A pair of nodes v_i and v_j are called **neighbors** or **adjacent** if they are within each others' transmission range (e.g., v_1 and v_3 in Fig. 1). Similarly, a pair of non-adjacent nodes v_i and v_j are called **adjoining** if their transmission disks *TD*(i) and *TD*(j) intersect (e.g., v_2 and v_5 in Fig. 1).

Now, consider what happens to the connectivity of the network when a node, say v_1, (in Fig. 2) is moved slightly. It is likely that the connectivity will remain the same, and thus not induce any chances in the Unit Disk Graph. However, if we continue to move the node in some direction two kinds of events can occur. A node that was within the transmission region of v_1 at the beginning may fall outside the range. For example, if node v_1 is moved upwards in the y-direction, node v_4 will fall outside the transmission region of v_1. We call such event an **excluding event**. If the node continues to move further upwards in the y-direction, node v_5, which was outside the transmission range of v_1 at the start, will appear within the range. We call this type of event an **including event**. This observation leads us to model free region for nodes as follows in Definition 1.

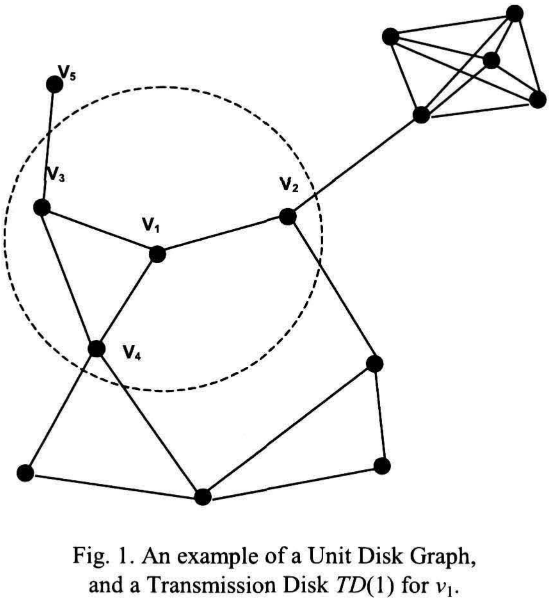

Fig. 1. An example of a Unit Disk Graph, and a Transmission Disk *TD*(1) for v_1.

* Howard R. Hughes College of Engineering, University of Nevada, Las Vegas, NV 89154-4026, e-mail: laxmi@cs.unlv.edu, nrongra@gmail.com, Henry.Selvaraj@unlv.edu, matt@cs.unlv.edu

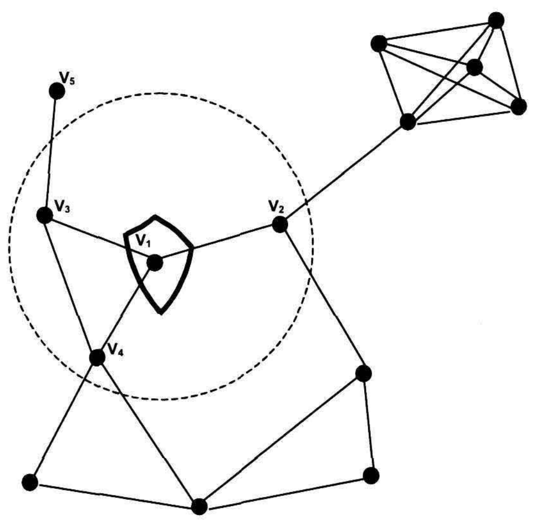

Fig. 2. The free region of node v_1.

Definition 1. The **free region** of a node v_i, denoted by $FR(i)$, is the open connected set of points in its neighborhood that preserves the connectivity of the network, that is, the area in which the node can move freely without altering the connectivity of the Unit Disk Graph.

A free-region $FR(i)$ of a node v_i is called **maximal** if it is not a proper subset of any other free-region of v_i. Figure 2 illustrates a free-region for node v_1. The region bounded by thick edges is the free region. It can be verified that this free-region in Fig. 2 is also maximal.

2. Preliminaries

Consider the **outer circle** $OC(i)$ of radius $2r$ centered at node v_i as shown in Fig. 3. The outer circle together with the transmission circle form the **annulus** $ANL(i)$ induced by node v_i. Sensor nodes lying within the transmission disk $TD(i)$ are referred to as the **inner nodes** of v_i. Similarly, nodes lying between the transmission circle and the outer circle are referred to as **outer nodes** of v_i. In Fig. 3, there are three inner nodes (v_2, v_3, v_4) and five outer nodes.

The notion of a free-region can be captured in terms of the transmission disks of (i) node v_i, (ii) its inner nodes, and (iii) its outer nodes. The region of intersection of transmission disks of inner nodes gives the region in which node v_i can be relocated without disconnecting with its adjacent nodes, even though some new nodes may become adjacent. This region which we call **in-free-region** $IFR(i)$ (Fig. 4) can be expressed in terms of the intersection of transmission disks as given in equation (1).

$$IFR(i) = \bigcap_j TD(j) \tag{1}$$

where $j = i$ or v_j is a neighbor of v_i.

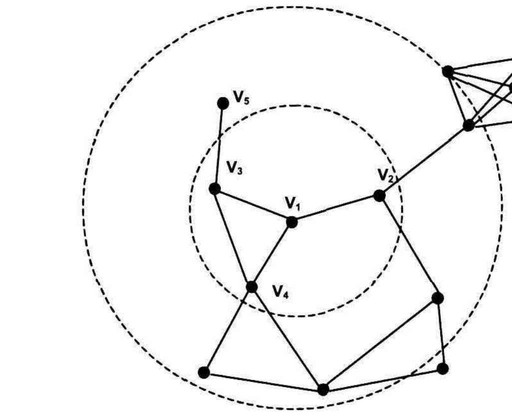

Fig. 3. Showing annulus, inner, and outer nodes.

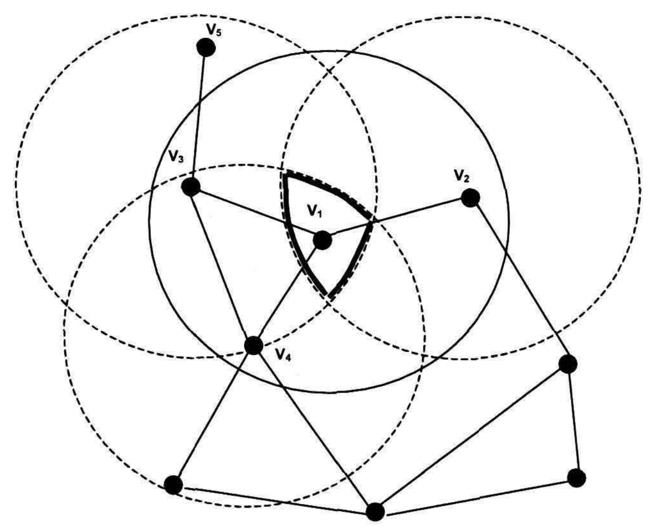

Fig. 4. Formation of in-free region (IFR) for node v_1.

The portion of the transmission disk $TD(i)$ that overlaps with the transmission disks of its out-bound nodes is referred to as **fringe region**. The region obtained by removing fringe regions from $TD(i)$ is called **out-free region** (see Fig. 5). The out-free region can be formally expressed as

$$OFR(i) = TD(i) - \bigcup_j TD(j) \tag{2}$$

for all outer nodes v_j of nodes v_i.

It is noted that as long as a node stays within its out-free-region, the set of nodes that were outside its transmission range at the initial position will continue to remain outside. The free region $FR(i)$ of node v_i is given by the intersection of its in-free region and out-free region. In fact, the (maximal) free region shown

in Fig. 2 is the intersection of free regions shown in Fig. 4 and Fig. 5. Free region $FR(i)$ of node i can be expressed as shown in equation (3).

$$FR(i) = \bigcap(OFR(i), IFR(i)). \tag{3}$$

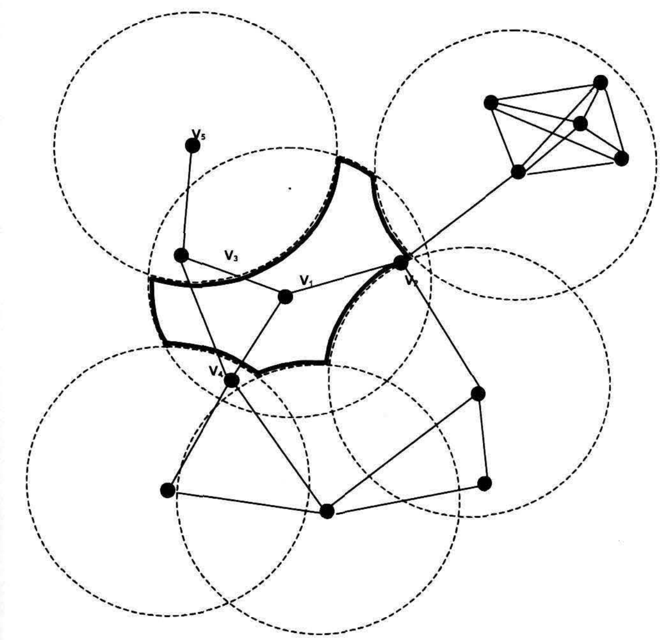

Fig. 5. Formation of the out-free region (OFR) of node v_1.

Remark 1. Both $OFR(i)$ and $IFR(i)$ are bounded regions whose boundary consists of **arc-chains**. Such regions are essentially special polygons whose edges are circular arcs and we refer to them as ***arc-gons***.

3. Incremental algorithm

An algorithm for computing $OFR(i)$ for a node v_1 can be developed by using an incremental approach in which outer nodes are processed one at a time. The outer nodes are first angularly sorted about v_i. Let the angularly sorted list of outer nodes be $v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_k}$. These nodes are processed one at a time in the order they appear in the sorted list. Initially, the transmission disk $TD(i)$ is taken as $OFR(i)$, whose boundary consists of just one arc, namely the transmission circle $TC(i)$. The region of intersection between $OFR(i)$ and the disk $TD(i_1)$, denoted as $IR(i_1)$ is subtracted from $OFR(i)$ to account for the cover of node v_i. The second node is processed similarly to update $OFR(i)$. The process of updating $OFR(i)$ incrementally, one node at a time, is continued for all outer nodes. At the j-th stage, the intersection region $IR(i_j)$ between $OFR(i)$ and $TD(i_j)$ is subtracted from the running $OFR(i)$ to account for the cover from node v_{i_j}.

It is noted that at the j-th stage the arc-gon representing the running $OFR(i)$ can have at most $2j$ arcs. When the j-th out-bound node is processed, transmission disk $TD(i_j)$ may not intersect with the running $OFR(i)$, for a certain class of node distributions. On the other hand, for some other class of node distributions, the transmission disk $TD(i_j)$ could possibly intersect with $O(k)$ arcs of the arc-gon representing the boundary of the running $OFR(i)$, where k is the number of out-bound nodes of v_i. The arcs of $OFR(i)$ that lie completely inside $TC(i_j)$ are called interior arcs. To update $OFR(i)$, interior arcs and intersecting arcs are removed from it.

The arcs of $OFR(i)$ that lie completely inside $TC(i_j)$ are called interior arcs. To update $OFR(i)$, interior arcs and intersecting arcs are removed from it. Up to three new arcs are formed by the intersection: one each corresponding to the intersecting arcs and one is the arc of $TC(i_j)$ between the intersecting arcs. The newly formed arcs are inserted into $OFR(i)$ to update it. A formal sketch of the algorithm is listed as the INCR-OFR Algorithm in Fig. 6.

Input: **a.** Sensor nodes $v_1, v_2, \ldots, v_n$
b. Transmission radius r
c. Integer i, $1 \le i \le n$

Output: Array *Arc[]* and its size m representing $OFR(i)$

Step 1: **a.** Determine out bound nodes of v_i
b. Angularly sort out bound nodes of v_i
c. Let the sorted list be $v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_k}$

Step 2: // Let *Arc[2k]* be the array to record the arcs of $OFR(i)$
a. *Arc[0]= TC(0)*;
b. m. = 1; // Number of arcs in the arc-gon
c. for (int j = 1; j <= k; j++)
d. if ($TC(vj)$ intersects with arc-gon Arc[] of size m)
Update(*Arc[]*, *m*, $TC(vj)$)

Step 3: Output *Arc[]* and its size *m*

Update(int *Arc[]*, *int &m*, $TC(vj)$) {
a. Find the intersecting arcs a_1 and a_2
b. Let g be the number of interior arcs.
c. Remove interior and intersecting arcs from *Arc[]*
d. Determine the newly formed arcs b_1, b_2, and b_3
e. Insert b_1,b_2, and b_3 into *Arc[]*
f. $m = m - g + 1$;
}

Fig. 6. Incremental OFR algorithm (INC-OFR).

Lemma 1. *INCR-OFR Algorithm can be executed in $O(k^2)$ time, where k is the number of inner and outer nodes of the candidate node.*

Proof: The intersecting arcs of an arc-gon of size k and a circle can be found in $O(k)$ time by simply checking the intersection of the circle with each arc of the arc-gon. The interior arcs are precisely the arcs in the arc-gon lying between the intersecting arcs. Hence intersecting arcs and their count can be determined in $O(k)$ time. The removal of interior arcs and the insertion of new arcs can be done in $O(n)$ time. Hence the *Update*() function can be done in $O(k)$ time. Since the *Update()* function is called at most $O(k)$ time the total time of the INCR-OFR Algorithm is $O(k^2)$. □

An algorithm for computing the in-free region *OFR*(i) can be developed by following the incremental approach similar to the one used for computing the out-free region. We omit the detail and state it in the following lemma.

Lemma 2. *The in-free region IFR*(i) *of node i can be computed in $O(k^2)$ time.*

Intersection of In-Free Region (IFR) and Out-Free Region (OFR)

Both OFR and IFR are special polygons whose edges are circular arcs and we refer to them as **arc-gons**. Computing the intersection of two arc-gons can be done by using the standard tools of computational geometry [3, 6]. When we examine the overlay of two arc-gons, the arcs of one arc-gon may intersect with arcs of the other arc-gon. We call one of the arc-gons the red arc-gon and the other the blue arc-gon. Let m and n be the number of arcs in the red and blue arc-gons, respectively.

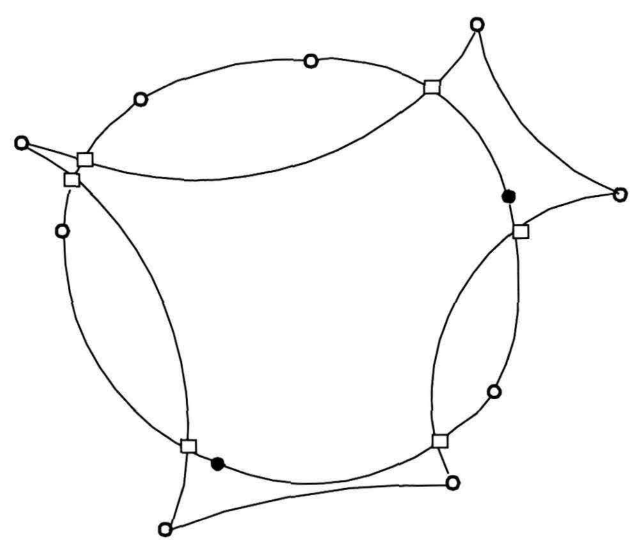

Fig. 7. Incremental OFR algorithm (INC-OFR).

Let k be the number of intersection points between the arcs of the red and the blue arc-gons. In terms of the overlay of the two arc-gons, the set of vertices of arc-gons and the vertices formed by the intersection of arcs can be distinguished into three sets: A vertex of one arc-gon that lies in the interior of the other arc-gon is called an **internal vertex**. Similarly, a vertex of one arc-gon that lies in the exterior of the other arc-gon is called an **external** vertex. Lastly, vertices formed by the intersection of arcs are referred to as **cross** vertices. Figure 7 shows the overlay of two arc-gons where the internal vertices are drawn filled, the external vertices are drawn unfilled and the cross vertices are drawn as little squares.

Definition 2. In the overlay of two arc-gons, the maximal arc-chain of one arc-gon that lies completely inside the other arc-gon is referred to as an **interior arc-chain**.

It may be noted that the end vertices of a maximal interior arc-chain are both cross vertices.

Observation 2. The boundary of the intersection of two arc-gons consists of a sequence of interior arc-chains.

Based on Observation 2, the intersection of the two arc-gons (the red and the blue arc-gons) can be determined by traversing the boundary of the red arc-gon and the blue arc-gon in an alternating manner by following a carefully formulated strategy.

The strategy is to traverse only along the interior arc-chain in each arc-gon; the traversal starts from any cross-vertex. From the initially picked cross-vertex, the traversal proceeds along the boundary that corresponds to the internal arc-chain. When the next arc-vertex at the end of the currently traversed arc-chain is encountered, the traversal switches to the boundary of the other arc-gon (say, the red arc-gon). This alternating traversal continues until the starting cross vertex is reached.

At the start of the traversal it is necessary to check whether or not the next vertex is inside the other arc-gon to determine the interior arc-chain. Point inclusion checking in simple polygons is a well known technique in computational geometry [3, 6]. We can use a similar technique to check point inclusion in an arc-gon which can be accomplished in $O(m + n)$ time. After the first interior arc-chain is determined, it is not necessary to check for point inclusion to determine the other interior arc-chain. This is due to the fact that the maximal interior arc-chain occurs in an alternating manner in red and blue arc-gons. If one of the interior arc-gon is in the red arc-gon then the next interior arc-gon occurs on the boundary of the blue arc-gon. A for-


```
Input: Red and blue arc-gons arc-gon-r and arc-gon-b of size m and n,
       respectively. The arcs in arc-gon-r and arc-gon-b are available in arrays.

Output: Arc-gon-i, representing the intersection of arc-gon-r and arc-gon-b.

Step 1: /* Determine cross vertices */
  a. For each arc arc-i in arc-gon-r do
  b.   For each arc arc-j in arc-gon-b do
  c.     If arc-i and arc-j intersect
  d.       Set cross-vertex W to the intersection of arc-i and arc-j
  e.       Split arc-i, arc-j at W
  f.       Record the references of arc-i and arc-j in record of W
  g.       Insert W in arc-i and arc-j

Step 2: /* Find the starting inner arc-chain */
  a. Traverse the boundary of arc-gon-r until a cross vertex cv_i is found
  b. Select the interior arc-chain arc-chain-i incident at cv_i;
  c. arc-gon-i = arc-chain-i;

Step 3: /* construct intersection arc-gon arc-gon-i */
  a. While (the other end point of arc-chain-i is not cv_i ) do
  b.   Set arc-chain-i to the next interior arc-chain at the end of the
         current arc-chain;
  c.   arc-chain-i = arc-gon-i ∪ arc-chain-i;

Step 4: Output arc-gon-i as the required intersection.
```

Fig. 8. The Red/Blue Intersection Algorithm (RBIA).

mal sketch of the algorithm is listed in Fig. 8 as the Red Blue Intersection Algorithm (RBIA).

Lemma 3. *Red-Blue Intersection algorithm executes in $O(n^2)$ time.*

Proof: Assume without loss of generality that $m = n$. Step 1 has two nested loops each of which executes $O(n)$ time and hence the time for Step 1 is $O(n^2)$. Step 2 takes $O(n)$ time. In Step 3, the traversal is done only on the boundary of the interior arc-chain and hence this step takes $O(n)$ time. Thus the total time complexity is $O(n^2)$.

From Lemma 1, Lemma 2, and Lemma 3, we find that in-free region, out-free region, and their intersection can be computed in $O(n^2)$ time. Hence we have the following theorem. □

Theorem 1. *Free region of a sensor node can be computed in $O(n^2)$ time.*

4. Conclusion

We presented a characterization of the free-region of a sensor node. The notion of free-region is use for relocating sensor nodes without compromising connectivity of the network. We presented a centralized algorithm for constructing the free-region of a sensor node in a network which executes in $O(k^2)$ time where k is the number of inner and outer nodes of the candidate node. We have established [7] that the problem of computing free region of a sensor node has lower bound $\Omega(n \log n)$. It would be interesting to design efficient approximation algorithms for constructing the free-region. We have made some progress in this direction and the result will be reported in the future.

References

[1] Bose P., Morin P., Stojmenovic I., Urrutia J., *Routing with Guaranteed Delivery in Ad Hoc Wireless Networks, Wireless Networks*, Vol. 7, Issue 6, 2001, pp. 609–616.

[2] Chakrabarty K., Iyengar S., Qi H, Cho E., *Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks*, IEEE Transactions on Computers, Vol. 51, Issue 12, 2002, pp. 1448–1453.

[3] de Berg M., Kreveld M. van, Overmars M., Schwarzkopf O., *Computational Geometry: Algorithms and Applications*, Springer, 1997.

[4] Fang Q., Gao J., Guibas L., *Locating and Bypassing Routing Holes in Sensor Networks*, Infocom 2004, Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, Vol. 4, 2004, pp. 2458–2468.

[5] Kar K.m Banerjee S., *Node Placement for Connected Coverage in Sensor Network*, Proceedings of WiOpt, 2003.

[6] O'Rourke J., *Computational Geometry in C*, 2nd edition, Cambridge University Press, 1998.

[7] Rongratana N., *Relocation and Deployment of Sensor Nodes*, M.Sc. Thesis, School of Computer Science, University of Nevada, Las Vegas, 2007.

[8] Sibley G.T., Rahimi M.H., Sukhatme G.S., *Robomate*: *A Tiny Mobile Robot Platform for Large Scale Sensor Network,* Proceedings of the IEEE International Conference on Robotics and Automaton (ICRA), 2002.

[9] Wang G., Cao G., Porta T., Wensheng Z., *Sensor Relocation in Mobile Sensor Networks*, Proceedings of IEEE INFOCOM, 2005, pp. 2302–2312.

[10] Wang X., Xing W., Zhang Y., Lu C., Pless R., Gill C., *Integrated Coverage and Connectivity Configuration in Wireless Sensor Network*, Proceedings of SenSys'03, 2003, pp. 28–39.

[11] Zhang H., Hou J., *Maintaining Sensing Coverage and Connectivity in Large Sensor Network*, Technical Report, UIUC, UIUCDCS-R-2003-2351, 2003.

[12] Zhao F., Guibas L., *Wireless Sensor Networks*, Morgan Kaufmann Publishers, 2004.

Received February 2, 2008