A Novel Data Structure for Unit Disk Graphs

J. Li, X. Liang, H. Selveraj, V. Muthukumar, and Laxmi P. Gewali*
School of Computer Science
University of Nevada, Las Vegas
{jianhong, xiaojunl, selvaraj, venkim}@egr.unlv.edu, laxmi@cs.unlv.edu

Abstract:
For constructing routes in mobile ad-hoc networks (MANET) and sensor networks it is highly desirable to perform primitive computations locally. If a network can be represented in the doubly connected edge list (DCEL) data structure then many operations can be done locally. However, DCEL data structure can be used to represent only planar graphs. In this paper we propose an extended version of DECL data structure called ExtDCEL that can be used for representing non-planar graphs as well as their planar components. The proposed data structure can be used to represent geometric networks in mobile computing that include unit disk graphs, Gabriel graphs, and constrained Delaunay triangulations. We show how the proposed data structure can be used to implement hybrid greedy face routing algorithm in optimum O(m) time, where m is the number of edges in the unit disk graph. We also report on the implementation of several routing algorithms for mobile computing by using the proposed data structure.

1. Introduction

Research and development problems dealing with mobile ad-hoc networks (MANET) and sensor networks has been considered by several researchers in recent years [1-5]. The nodes in both types of networks are low-powered computing elements equipped with wireless transmission device of finite range. There is no fixed infra structure to connect such nodes. Nodes within the transmission range can exchange messages directly, and nodes outside of their transmission range can possibly exchange messages via a sequence of intermediate relay nodes. The process of selecting an ordered sequence of intermediate nodes to exchange messages between a source node S and a target node T is called *routing*. In both type of networks, nodes can be in either active or inactive state, and in MANET they can change their position. Due to node mobility and the possibility of their state change, the connectivity of wireless network changes with the progress of time in both types of networks. Any routing algorithm designed for ad-hoc networks can use only small limited amount of processing time and memory space per node. Furthermore, most primitive computations should be done locally. One of the main objectives in

* Corresponding Author

mobile computing is to determine global properties of ad-hoc networks by performing local computation on neighboring nodes.

Network models from computational geometry have been used with success in designing mobile ad-hoc networks. Delaunay triangulations, Gabriel graphs, unit disk graphs, relative neighborhood graphs, and Yao graphs are some of the geometric structures used for constructing location based routes in mobile computing [8]. For investigation and the actual implementation of routing algorithms it is necessary to represent the networks in a suitable data structure. One of the widely used data structures in computational geometry for representing geometric graphs is the doubly connected edge list (DCEL) structure [7]. However, DCEL can be used to represent only planar graphs.

In this paper, we propose a data structure called Extended Doubly Connected Edge List (ExtDCEL) that can be used to represent any non-planar geometric graph as well as its planar components. This data structure is very useful for implementing various location-based routing algorithms in mobile computing. In Section 2, we present a brief review of location based routing algorithms. In Section 3, we describe the proposed ExtDCEL data structure and show how it facilitates quick construction and quick update of routes in both planar and non-planar graphs and present some actual implementation results. The implementation includes greedy routing, standard face routing, hybrid greedy-face routing, and improved face routing.

2. Review of Geometric Routing Algorithms

The routing problem in MANET is to construct a path between a source node S and a destination node T that are not within the transmission range. All consecutive nodes in the constructed S-T-path must be within the transmission range so that the path can be used to route messages from S to T. Most of the geometric algorithms are essentially location based. In location based routing algorithms it is assumed that the sender knows the location of the destination node within certain accuracy. This is a valid assumption if the nodes are assumed to have access to global positioning service. These algorithms can be categorized into three types: (i.) greedy routing, (ii.) face routing, and (iii.) hybrid routing. We assume that the transmission ranges of all nodes are identical and equal to some constant value. Without loss of generality the transmission range is taken as unity. The location based routing problem can be formally defined as follows.

Location Based Routing Problem: *Given (i) a set V of mobile nodes equipped with wireless devices, (ii) a start node S, and (iii) a target node T, construct a route connecting them that can be used to transmit exchange messages.*

A straightforward and intuitive model of ad-hoc network is obtained by connecting all pair of nodes lying within the transmission range. Such a network is called the ***unit disk graph*** **(UDG).** The unit disk graph is not necessarily planar and could be very dense if the number of nodes per unit area is large. Figure 1 shows the unit disk graph for 16 nodes.

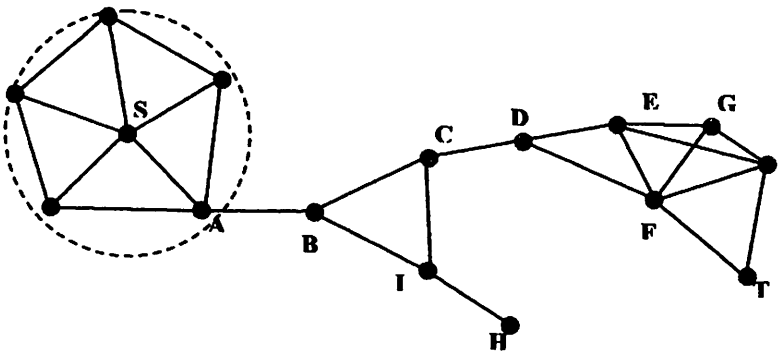


Figure 1: Illustrating the Unit Disk Graph (UDG)

2.1 Greedy Routing

One of the simplest algorithms for constructing a source target path is the greedy most forward routing algorithm [8]. In this approach, the source node S forwards the message to the node that is closest to the target node T. This is a pure localized algorithm and works well if the number of nodes per unit area is very large. However, for some node distribution, this approach can get trapped in a local minimum and the algorithm fails to construct the route even when several routes connecting source to destination exist. For example, in Figure 1, source node S forwards message to node A, which is the node closest to the target node T. Next, node A forwards the message to node B which is the node closest to node T. Continuing this greedy forwarding strategy, the message reaches node H, from where there is no neighboring node that makes positive progress towards the target node and the message gets trapped there. Greedy most forward algorithm constructs routs successfully if the number of nodes per unit area is very large. These algorithms are relatively simpler to implement.

2.2 Face Routing

The technique that is always successful in constructing a route (if a route exists) is the face-routing algorithm [1, 4-5]. The first idea of face routing technique was reported in the paper dealing with compass routing [1]. For the

applicability of the face routing algorithm, the network must be a planar graph. Fortunately, geometric graphs such as the Gabriel graphs (GG) and relative neighborhood graph satisfy planarity property and efficient algorithms are known for their construction. The notion of Gabriel graph can be easily visualized in term of empty disk test. For a pair of nodes u and v, let $D(u,v)$ denote the disk having diameter the line segment with end points at u and v. Nodes u and v are connected by an edge in the Gabriel graph if $D(u,v)$ does not contain any other node. If the size of the empty disk is restricted to be no more than the transmission range then we get Constrained Gabriel Graphs (CGG). CGG are known to capture the connectivity of mobile network. Figure 2 shows an example of a Constrained Gabriel graph. It is straightforward to observe that Gabriel graphs can be constructed locally by only examining the pair of nodes that lie within the transmission range.

The face routing algorithm constructs a route connecting source node S to node T by exploring the faces of the graph intersected by the line segment connecting the source and the destination nodes. The algorithm constructs the path by processing faces one by one in the order they occur along the source destination line segment. It is known that face routing algorithm guarantees to yield a route if a route exist [1]. Figure 2 illustrates the route connecting nodes S and T in a Gabriel graph constructed by face routing approach. The constructed route is drawn as shaded segments. As seen in the figure the route is constructed by using portions of the boundaries of the faces intersected by the line segment connecting S and T. If the planar graph has faces with large number of boundary edges then the face routing algorithm may yield a route with large number of hops.

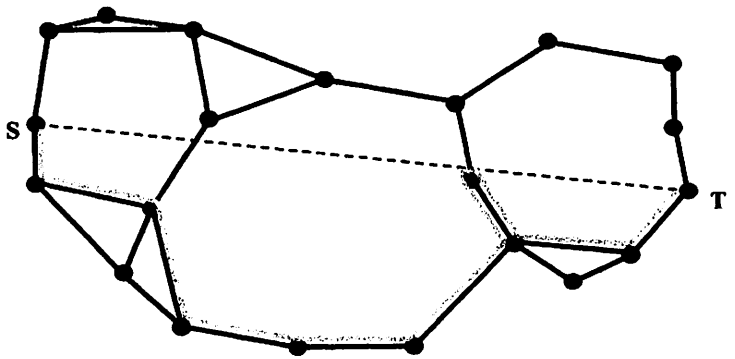


Figure 2: Illustrating the Gabriel Graph

If the Gabriel graph has a very large size face then the face routing algorithm may yield a route with large number of hops. To address this issue, several

variations of face routing algorithms have been proposed [4,5]. In one such variations, the search area is bounded by an ellipse, whose parameter is determined by the length of the shortest path from S to T. The length α of the shortest path between S and T is estimated initially as twice the length of the straight line segment S and T. If a route is contained within the initially chosen ellipse, then the algorithm becomes successful. Otherwise, the size of the ellipse is made bigger by doubling the estimated shortest path α. This doubling process continues until the search becomes successful. It may be noted that when the search is not successful at any stage, the algorithm returns back to the initial starting point S. It is shown [4,5] that this bounded adaptive face routing algorithm finds a route in $O(\alpha_0)$ time, where α_0 is the number of edges in the shortest path connecting S to T.

2.3 Hybrid Routing

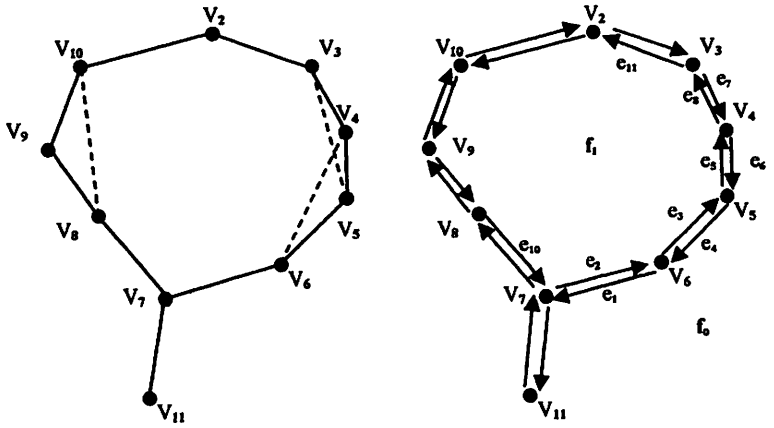
As observed above, the greedy routing generates short length routes for dense network and could get trapped in a local minimum on sparse graphs. On the other hand, face routing is always successful for route generation if a route exit, but could generate long length route if the underlying planar graph contains large size faces. The reason face routing generates long length route is that the many short edges are eliminated when a planar graph such as the Gabriel graph is extracted from the unit disk graph. To address this issue hybrid algorithms that switch back and forth between face routing state and greedy routing state have been developed [8]. The algorithm works by representing the network in both Gabriel graph and unit disk graph. The hybrid routing algorithm starts by executing the greedy most forward algorithm on the unit disk graph. If the greedy strategy is trapped in a local minimum then the algorithm uses face routing in the Gabriel graph to escape from the trap and resumes the greedy method in the unit disk graph. Since all nodes of the unit disk graph are also in the Gabriel graph the algorithm can always find an escaping face from the trapping node at any local minimum. This approach is known to work fairly well for large class of mobile networks.

3. Extended Doubly Connected Edge List (ExtDCEL) Data Structure

One of the commonly used data structures for representing planar graphs is the doubly connected edge list (DCEL) [3,7]. DCEL is suitable for updating the graph efficiently when there is change in nodes or edges or in connectivity. In DCEL representation if a node changes its position the resulting change in the connectivity can be done locally. But the network model used in mobile computing may not necessarily be planar.

For the representation of non-planar geometric graphs, we propose an enhancement of doubly connected edge list data structure called Extended

Doubly Connected Edge List (ExtDCEL). The proposed ExtDCEL data structure can be used to represent the non-planar graphs UDG as well as a planar approximation of UDG such as, the constrained Gabriel graph. As in DCEL data structure, each edge of the graph is represented by a pair of oppositely directed half-edges which are twin of each other. Reference [7] gives the detail of DCEL data structure. To obtain ExtDCEL, we add two additional information in the standards DCEL: (i) for each node v_i, we maintain a list of neighbor nodes that are within the transmission range; and (ii) for each half edge e_i, we maintain the jump edge e_{jump} that can be used to skip certain number of nodes in the face on which e_i is incident. The neighbors of a node v_i can be used to construct the unit disk graph and jump edges can be used to route the path around the face quickly. Figure 3(a) shows the unit disk graph of 10 mobile nodes with the indicated transmission range. In the graph non-Gabriel edges are drawn as dashed line segments.



(a) Unit Disk Graph Graph (b) ExtDCEL Representation

Figure 3: Illustrating ExtDCEL Data Structure

ExtDCEL represents the graph by maintaining records for nodes, half-edges, and faces. A connected non-planar graph such as UDG can be viewed to consist of a connected planar component and a few extra edges. Each edge in the planar component is incident on a face. In term of the counterclockwise traversal of a face, each half-edge of the planar component has unique half-edge as its predecessor and a unique half-edge as its successor. The record of a half-edge includes this relationship and other information. Specifically, the record for each half-edge e_i contain entries for indexes of (i) start node of e_i, (ii) twin half-edge of e_i, (iii) next half-edge of e_i, (iv) previous half-edge of e_i, (v) the face incident

on e_i, and (vi) jump-edge of e_i. Similarly, the record for each node v_i include entries for (i) node id, (ii) index of the half edge incident on v_i, (iii) coordinates of v_i, and (iv) the list of neighbors of v_i. A face-record contains entries for its id and the index of a half-edge on its boundary. Table 1, Table 2, and Table 3 show the partial vertex, edge, and face records for the graph shown in Figure 3.

Vertex	Coordinates	Incident-Edge	Neighbors
V_4	(x4, y4)	e_8	V_3, V_5
V_5	(x5, y5)	e_5	V_4, V_6
V_6	(x6, y6)	e_3	V_5, V_7
V_7	(x7, y7)	e_2	V_6, V_8, V_{11}

Table 1: Listing of Vertex Records

Half-Edge	Twin	Next	Prev	Inc Face	Start Vertex	Jump Edge
e_2	e_1	e_3	e_{10}	f_1	V_7	e_7
e_3	e_4	e_5	e_2	f_1	V_6	e_{11}
e_5	e_6	e_8	e_3	f_1	V_5	--

Table 2: Listing of Half-Edge Record

Face Index	Inc Edge
f_0	e_1
f_1	e_2

Table 2: Listing of Half-Edge Record

The proposed data structure was used to implement a variety of routing algorithms that include the face routing algorithm, greedy most forward algorithm, hybrid greedy face routing, and improved face routing algorithm. The implementation is done in the Java programming language. The performance of the algorithms was examined by executing them on several test data. Some test data were generated randomly and some were designed to have very large size faces. Snap-shots of the output generated by the implemented program are as shown in Figures 4-6. The ExtDCEL data structure can be used to efficiently construct the standard face route and the hybrid greedy face route. These algorithms are listed as Standard FaceRouting (Subsection 3.2) and GreedyFaceRouting (Subsection 3.3).

3.1 Time Complexity Analysis

The Standard Face Routing Algorithm listed above is the description of the face routing algorithm given in [1] by using the proposed data structure. The detail sketch of time complexity analysis of the face routing algorithm is given

in [1]. It is established in [1] that the time complexity of face routing algorithm is *O(n)*, where *n* is the number of vertices in the Gabriel graph. The time complexity of hybrid Greedy Face routing algorithm implemented by using ExtDCEL data structure is stated below.

***Theorem 1**: The time complexity of GreedyFaceRouting algorithm is O(m), where m is the total number of vertices in the unit disk graph.*

Proof: From the vertex record (Table 1) all neighbors of a vertex can be found in O(d) time, where d is the degree of that vertex. By examining the neighbors of a vertex v, the vertex nearest to the target node T can be found within the same time. Hence, one execution of ***mostForward (startVertex, T)*** function takes O(d) and consequently each execution of the while loop of Step 2.1 takes O(d) time. If k edges are traversed in Step 2.2 by face routing then the time for that traversal is O(k). Since the traversal by the greedy algorithm is done only in the forward direction, each vertex is processed only a constant number of time. If r vertices are processed in total (most forward and face routing) then the total time is O(rd'), where d' is the average degree of processed vertices. It is observed that the sum total of the degree of all vertices is equal to twice the total number of edges m in the graph. Hence the total time for the algorithm is O(m).

3.2 Algorithm Standard FaceRouting

Step 1: i) Let **S, T** be the start and target vertices, respectively;
ii) Let ***startEdge*** be the half edge incident at **S**;
iii) ***currEdge = startEdge***; output ***currEdge***;
iv) ***currEdge = currEdge.next; f = currEdge.incFace;***
v) startEdge = currEdge;

Step 2: while (***currEdge.startVertex*** != T){
// Scan the boundary of face f to identify the furthest edge of face f
// that intersects with segment **ST**.
intPoint = Null;

Step 2.1: ***while (currEdge != firstEdge)*** {
if (ST intersects with currEdge) {
if (intPoint == Null) {
exitEdge = currEdge;
intPoint = intersection(ST, currEdge);
}
else {
exitEdge = currEdge;
newIntPoint = intersection(ST, currEdge);
intPoint = furthest(intPoint, newIntPoint);
}

```
                    }
                    currEdge = currEdge.next;
                } // end Step 2.1
                //output edges of face from startEdge to exitEdge
                currEdge = startEdge;
Step 2.2        while (currEdge != exitEdge){
                        output currEdge;
                        currEdge = currEdge.next;
                } //end Step 2.2
                // switch to the next face
                currEdge = currEdge.twin;nstartEdge = currFace;
                f = currEdge.incFace; currEdge = currEdge.next;
                output curredge;
        } // end Step 2
```

3.3 Algorithm GreedyFaceRouting

```
Step 1:     i)  Let S, T be the start and target vertices, respectively;
            ii) startVertex = S;
            iii) endVertex = mostForward(startVertex,T);
Step 2:     while (startVertex != T) {
                endVertex = mostForward(startVertex,T);
Step 2.1:       while (endVertex != Null) {
                    output segment(startVertex, endVertex);
                    startVertex = endVertex;
                    endVertex = mostForward(startVertex,T);
                }//end Step 2.1
Step 2.2;       if (endVertex == Null)
                {
                    (i) Use standard face routing algorithm to escape from
the
                        local minima and identify new startVertex.
                    (ii) endVertex = mostForward(startVertex, T);
                } //end Step 2.2
        } // end Step 2
```

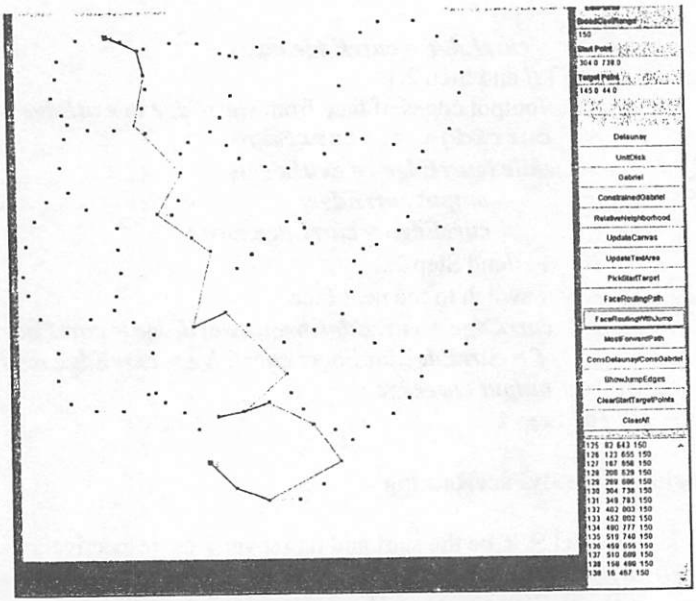


Figure 4: Route Generated by Improved Face Routing in Gabriel Graph

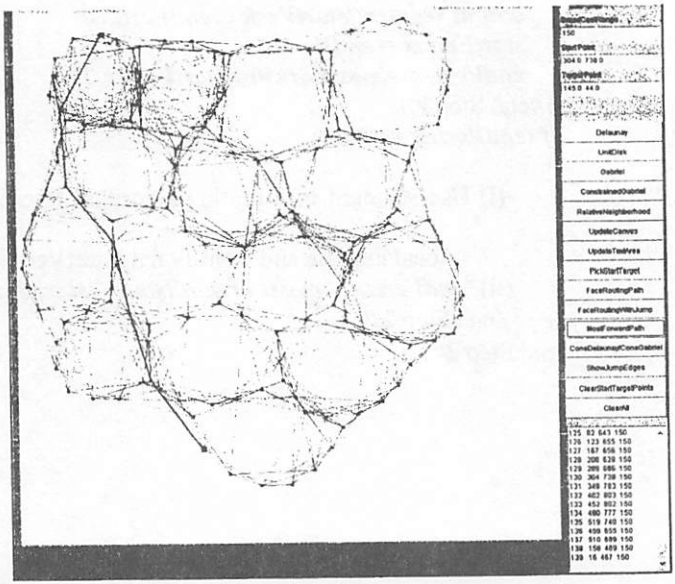


Figure 5: Route Generated by Greedy Routing in UDG

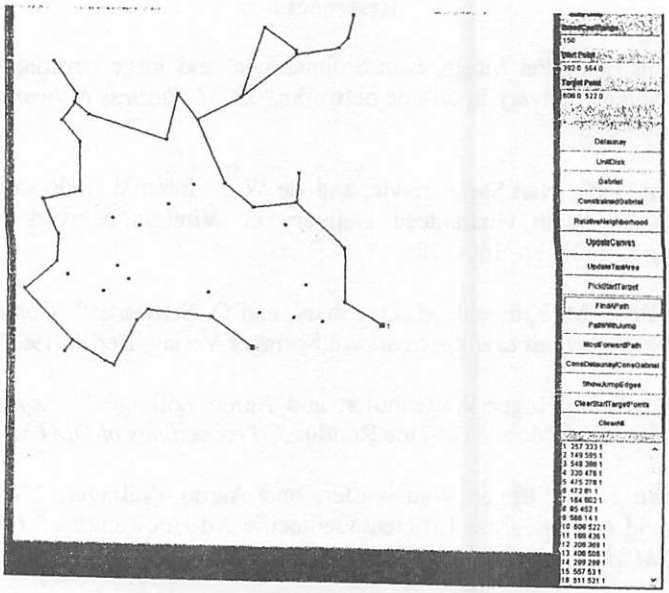


Figure 6: A Very Long Route Generated by Standard Face Routing Algorithm

4. Discussion

We presented an overview of geometric networks and routing algorithms applicable in mobile computing and sensor networks. We examined the data structures that can be used for representation and maintenance of planar graphs. We proposed an extension of doubly connected edge list data structure called extended doubly connected edge list (ExtDCEL) that can be used for implementing several routing algorithms in mobile computing and sensor networks. The proposed data structure is convenient for representing both planar and non-planar geometric graphs, and can be maintained with local computation. The proposed data structure is used to implement several location based routing algorithms for mobile computing. In particular, the hybrid greedy face algorithm can be implemented very easily by using ExtDCEL. Several complicated power aware routing algorithms have been considered in recent years. It would be interesting to investigate the applicability of ExtDCEL data structure for implementing power aware routing algorithms.

References

[1] Prosenjit Bose, Pat Morin, Ivan Stojmenovic, and Jorge Urrutia, ``Routing With guaranteed delivery in ad-hoc networks," *ACM Wireless Networks*,7, Nov. 2001, pp.609-616.

[2] Susanta Datta, Ivan Stojmenovic, and Jie Wu, ``Internal Node and Shortcut Based Routing with Guaranteed Delivery in Wireless Networks," *Cluster Computing* 5, 2002, pp. 169-178.

[3]M. de Berg, M. Krteveld, M. Overmars, and O. Schwarkoft, *Computational Geometry: Algorithms and Applications*, Springer Verlag, Berlin, 1997.

[4] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger, ``Asymptotically Optimal GeometricMobile Ad-Hoc Routing," *Proceedings of DIALM*, 2002.

[5] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger, ``Worst Case Optimal and Average Case Efficient Geometric Ad-Hoc Routing," *Proceedings of the ACM MOBIHOC*, 2003, pp. 267-278.

[6] J. Li, L. Gewali, H. Selvaraj, and V. Muthukumar, ``Hybrid Greedy Facw Routing for Ad-Hoc Sensor Networks,'' Proceedings of Euromicro Symposium on Digital System Design, 2004, pp. 574-578.

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

[8] Ivan Stojmenovic (Editor), *Handbook of Wireless Networks and Mobile Computing*, John Wiley and Sons, 2002.