Algorithm Utility Tool Kit for Random Disk Graph Thresholds

Evren Kaya

July 17, 2015

1 Introduction

Wireless networks are usually modeled as disk graphs in the plane. Given a set P of points in the plane and a positive parameter r, the *disk graph* is the geometric graph with vertex set P which has a straight-line edge between two points $p, q \in P$ if and only if $|pq| \leq r$, where $|pq|$ denotes the Euclidean distance between p and q. If $r = 1$, then the disk graph is referred to as *unit disk graph*. A *random geometric graph*, denoted by $G(n, r)$, is a geometric graph formed by choosing n points independently and uniformly at random in a unit square; two points are connected by a straight-line edge if and only if they are at Euclidean distance at most r, where $r = r(n)$ is a function of n and $r \to 0$ as $n \to \infty$.

We say that two line segments in the plane *cross* each other if they have a point in common that is interior to both edges. Two line segments are *non-crossing* if they do not cross. Note that two non-crossing line segments may share an endpoint. A geometric graph is said to be *plane* if its edges do not cross, and *non-plane*, otherwise. A graph is *planar* if and only if it does not contain K_5 (the complete graph on 5 vertices) or $K_{3,3}$ (the complete bipartite graph on six vertices partitioned into two parts each of size 3) as a minor. A *non-planar graph* is a graph which is not planar.

In order to test Theorems 1 and 4 from Biniaz et al. [1], we developed a computer program that serves as an algorithm utility kit for disk graphs. (An executable form of this program can be downloaded by clicking here (URL: $https://github.com/evrenkaya/UnitDiskGraphvWITHOUTGRAPHVIEW$) by clicking on "UnitDiskGraph.jar" and then "View Raw")

The following is a restatement of Theorems 1 and 4 from Biniaz et al. [1]:

Theorem 1 *Let $k \geq 2$ be an integer constant. Then, $n^{\frac{-k}{2k-2}}$ is a distance threshold function for $G(n, r)$ to have a connected subgraph on k points.*

Theorem 4 *$n^{-\frac{2}{3}}$ is a threshold for $G(n, r)$ to be plane.*

In this paper, we explain some of the key implementation details of our program and then show our tests conducted for each theorem.

2 Implementation Details

In this section, we describe the implementations of the key Java classes within our program.

Vertex

A vertex is a point in the unit square. Every vertex has x,y coordinates as doubles, a boolean variable to store visited state(used in Breadth First Search), and an adjacency list represented as an ArrayList<Vertex>.

Edge

A straight-line edge. Every edge has references to its endpoints, a boolean variable to keep track of whether it is intersecting with another edge, and a weight as a double.

UnitDiskGraph

A random disk graph. Stores all vertices as an ArrayList<Vertex>, all edges as an ArrayList<Edge>, and the current distance threshold as a double. The following methods in this class are now described in more detail:

createNewRandomVertices() - creates a new set of vertices uniformly at random in the unit square

createNewConnectedEdges() - compares the Euclidean distance between every pair of vertices(Pythagoras' Theorem) and if two vertices are at most the distance threshold apart, then a new edge is created with these two vertices as its endpoints

determineIntersectingEdges() - checks every pair of edges to see if they intersect at a point other than any common endpoints. This is done using Java's Line2D.linesIntersect() method

determineSuperFreeEdges() - checks every pair of free edges to see if there are any other vertices inside of the super free edge rectangular region

BreadthFirstSearch

A class representing the Breadth First Search algorithm. Contains a reference to the UnitDiskGraph object that it will be performing its search on. This class also stores all the connected components as an ArrayList<ArrayList<Vertex>>, in other words, a list of lists of vertices. The two methods of importance here are as follows:

getConnectedComponentWith(Vertex startingVertex) - starts breadth first search at the given vertex and returns the entire connected component containing this vertex as an ArrayList<Vertex>

determineAllConnectedComponents() - calls the above method for each vertex in the graph, keeping track of which connected components have been found already

3 Testing

In this section, we explain how we tested Theorems 1 and 4 using our algorithm utility program.

In order to test Theorem 1, we chose different values of n and k to input and then recorded the number of connected components with at least k points that appeared above, below and at the distance threshold function using a small value ε. Theorem 1 is verified if there is **at least one** connected component on k points **above** the distance threshold, and, there are **no** connected components on k points **below** the distance threshold. The following table shows our results for values of n ranging from 50 - 10000.(Note: In this table, a and b are variables that replace k and $2k-2$ respectively)

n	$a=k$	$b=2k-2$	ε	$r=n^{-(\frac{a}{b}+\varepsilon)}$	# Connected components with $\geq k$ points
50	5	8	-0.05	0.10546	3
50	5	8	0	0.08672	1
50	5	8	0.05	0.07131	1
50	10	18	-0.05	0.13838	1
50	10	18	0	0.11379	1
50	10	18	0.05	0.09357	0
50	30	58	-0.05	0.16075	0
50	30	58	0	0.13219	0
50	30	58	0.05	0.10871	0
100	5	8	-0.05	0.07079	3
100	5	8	0	0.05623	2
100	5	8	0.05	0.04466	0
100	10	18	-0.05	0.09747	4
100	10	18	0	0.07742	1
100	10	18	0.05	0.06150	0
100	30	58	-0.05	0.11628	1
100	30	58	0	0.09236	0
100	30	58	0.05	0.07336	0

500	5	8	-0.05	0.02806	16
500	5	8	0	0.02056	2
500	5	8	0.05	0.01507	0
500	10	18	-0.05	0.04320	16
500	10	18	0	0.03166	0
500	10	18	0.05	0.02320	0
500	30	58	-0.05	0.05481	5
500	30	58	0	0.04017	1
500	30	58	0.05	0.02944	0
1000	5	8	-0.05	0.01883	24
1000	5	8	0	0.01333	6
1000	5	8	0.05	0.00944	1
1000	10	18	-0.05	0.03043	23
1000	10	18	0	0.02154	5
1000	10	18	0.05	0.01525	0
1000	30	58	-0.05	0.03965	2
1000	30	58	0	0.02807	2
1000	30	58	0.05	0.01987	0
5000	5	8	-0.05	0.00746	85
5000	5	8	0	0.00487	9
5000	5	8	0.05	0.00318	0
5000	10	18	-0.05	0.01348	144
5000	10	18	0	0.00881	14
5000	10	18	0.05	0.00575	0
5000	30	58	-0.05	0.01869	1
5000	30	58	0	0.01221	8
5000	30	58	0.05	0.00797	0
10000	5	8	-0.05	0.00501	130
10000	5	8	0	0.00316	4
10000	5	8	0.05	0.00199	0
10000	10	18	-0.05	0.00950	277
10000	10	18	0	0.00599	21
10000	10	18	0.05	0.00378	0
10000	30	58	-0.05	0.01352	3
10000	30	58	0	0.00853	6
10000	30	58	0.05	0.00538	0

Table 1: Theorem 1 testing data. n is the number of points, a and b are variables that replace k and $2k-2$ respectively, and ε is a very small value used to vary the radius r. A random graph is generated for each row.

In order to test Theorem 4, we set $a = 2$ and $b = 3$ as constants and only varied n and ε. For each value of n, we recorded the number of intersecting edges within the disk graph that appeared above, below and at the distance threshold using a small value ε. Theorem 4 is verified if there **exists** intersecting edges **above** the distance threshold, and, there are **no** intersecting edges **below** the distance threshold. The following table shows our results for values of n ranging from 50 - 10000.

n	ε	$r = n^{-(\frac{2}{3}+\varepsilon)}$	# Intersecting edges
50	-0.001	0.07396	2
50	0	0.07368	2
50	+0.001	0.07339	0
50	-0.01	0.07662	0
50	0	0.07368	0
50	+0.01	0.07085	0
50	-0.05	0.08959	2
50	0	0.07368	2
50	+0.05	0.06059	0
100	-0.001	0.04663	0
100	0	0.04641	0
100	+0.001	0.04620	0
100	-0.01	0.04860	0
100	0	0.04641	0
100	+0.01	0.04432	0
100	-0.05	0.05843	0
100	0	0.04641	0
100	+0.05	0.03686	0
500	-0.001	0.01597	0
500	0	0.01587	0
500	+0.001	0.01577	0
500	-0.01	0.01689	0
500	0	0.01587	0
500	+0.01	0.01491	2
500	-0.05	0.02165	2
500	0	0.01587	0
500	+0.05	0.01163	0
1000	-0.001	0.01006	0
1000	0	0.00999	0
1000	+0.001	0.00993	2
1000	-0.01	0.01071	2
1000	0	0.00999	0
1000	+0.01	0.00933	0

1000	-0.05	0.01412	4
1000	0	0.00999	0
1000	+0.05	0.00707	0
5000	-0.001	0.00344	0
5000	0	0.00341	0
5000	+0.001	0.00339	0
5000	-0.01	0.00372	0
5000	0	0.00341	0
5000	+0.01	0.00314	2
5000	-0.05	0.00523	18
5000	0	0.00341	2
5000	+0.05	0.00223	0
10000	-0.001	0.00217	0
10000	0	0.00215	0
10000	+0.001	0.00213	0
10000	-0.01	0.00236	2
10000	0	0.00215	2
10000	+0.01	0.00196	0
10000	-0.05	0.00341	12
10000	0	0.00215	2
10000	+0.05	0.00135	0

Table 2: Theorem 4 testing data. A random graph is generated for each row. a and b are the constants 2 and 3 respectively.

4 Conclusion

From Table 1, we see that for many combinations of n, k, and ε, there exists many connected components on k points above the distance threshold and none below the threshold, thus verifying Theorem 1. From Table 2, we see that for high values of $n(\geq 1000)$, there exists intersecting edges above the distance threshold and that there are no intersecting edges below the threshold. This verifies that $n^{-\frac{2}{3}}$ is a distance threshold for $G(n, r)$ to be plane(i.e. Theorem 4).

Acknowledgments: I would like to thank both of my supervisors, Professor Anil Maheshwari and PhD student Ahmad Biniaz for assisting me in creating this report.

References

[1] A, Biniaz, E. Kranakis, A. Maheshwari and M. Smid. *Plane and Planarity Thresholds for Random Geometric Graphs.* 2015.