Linear Time Generation of Simulated Wireless Sensor Networks with Random Geometric Graphs

Luke Wood

March 5, 2018

1 Executive Summary

1.1 Introduction and Summary

Wireless Sensor Networks can be incredibly expensive to deploy and test which makes them an excellent candidate for simulated testing. Vlady Ravelomananana and Hichem Kenniche from the University of Paris first explored the concept of using random geometric graphs (RGGs) to attempt to model wireless sensor networks [2]. RGGs can be used as a relatively cheap way to gather valuable information about how wireless sensor networks can function and communicate. Through a series of reports I will be:

1. Generating RGGs on the geometries of a unit square, unit disc, and unit sphere
2. Color the generated graph in linear time using smallest vertex last ordering (TODO check)
3. Find the terminal clique in the generated RGGs
4. Find a selection of bipartite subgraphs producted by an algorithm for coloring

This report is the first of the series and will describe an implementation for a linear time algorithm to generate graphs consisting of N vertices with an average degree of A on the geometric topologies of: unit square, unit disc, unit sphere.

1.2 Programming Environment Description

The implementation of the algorithm used to gather the data supporting this report was gathered on a 15 inch Macbook pro 2017 with a 2.9 GHz Intel Core i7 processor and 16 GB of RAM.

Nodes	Expected Avg. Deg.	Real Avg. Deg.	Max Deg.	Min Deg.	Seconds
1000	64	57.568000	82	10	0.279913
5000	64	60.658400	87	15	0.840636
25000	64	62.504080	98	11	3.828407
50000	64	62.986840	100	20	8.938070
100000	64	63.258520	102	18	14.938207

Table 1: Data on graphs generated with the square topology

2 Reduction to Practice

2.1 Data Structure Design

2.2 Algorithm Description

In order to ensure that average degree of the nodes is close to the desired average degree we define a radius surrounding each node. The formulas to find the radius for each topology is derived from the equations found in the paper Bipartite Grid Partitioning of a Random Geometric Graph[1]. The formula used to find this radius varies for each graph tolopogy and can be found in the table displayed below:

Topology	Equation in Chen's Paper	Equation for Radius Derive from Chen's
Unit Square	$d(G) \approx N\pi r^2$	$r = sqrt(\frac{d(G)}{N\pi})$
Unit Disc		
Unit Sphere		

2.3 Algorithm Engineering

Originally I had a brute force algorithm that ran in $O(n^2)$ time. This quickly became problematic as the algorithm took upwards of 200 seconds to run on input size of only 12,000.

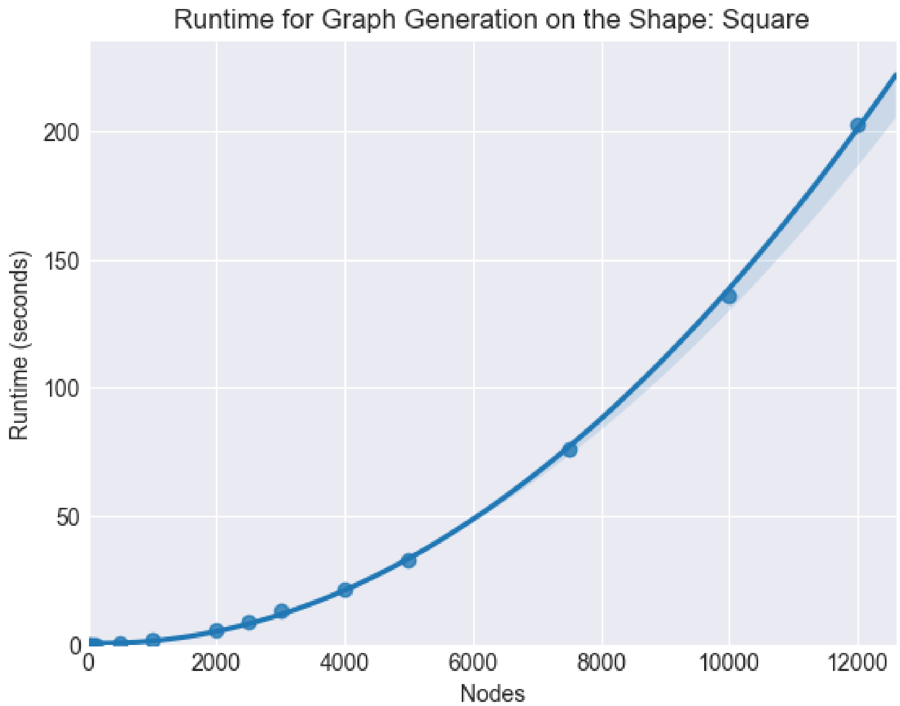

Figure 1: Runtimes of the $O(n^2)$ Algorithm

To fix this, I overhauled the algorithm to be $O(n)$. I did this by implementing the cell method described above in the Algorithm Description section as well as in Chen's paper Bipartite Grid Partitioning of a Random Geometric Graph[1].

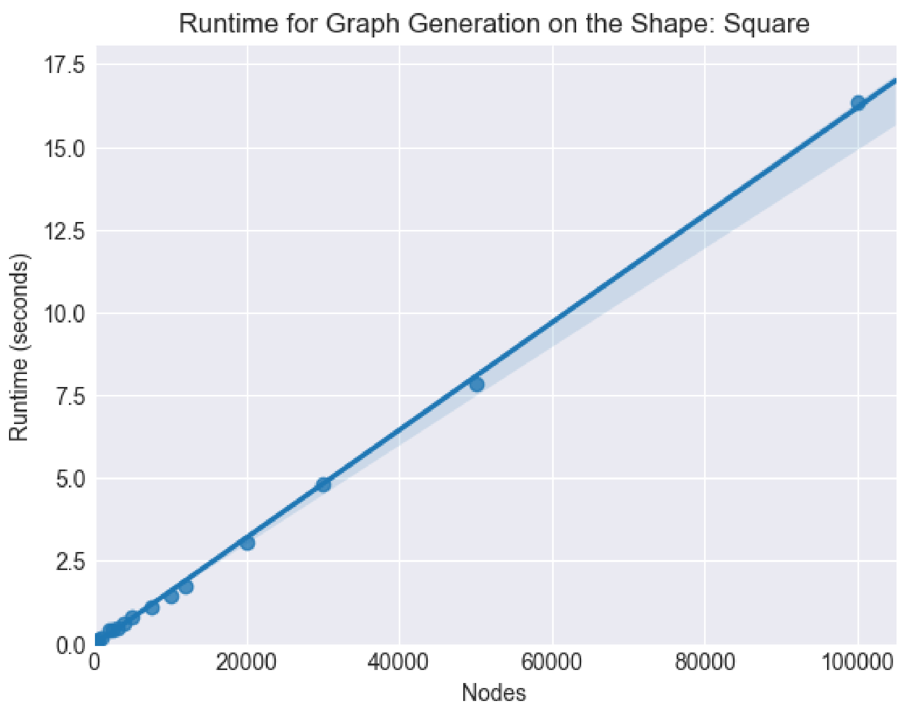

Figure 2: Runtimes of the $O(n)$ Algorithm

I have also included a table to compare some of the runtimes for the same size inputs below.

Nodes	$O(n^2)$	$O(n)$
1000	0.258400	0.657174
2000	0.382116	2.630040
3000	0.473916	5.874501
5000	0.831753	16.809004
10000	1.560216	65.398991

Table 2: Runtimes of the $O(n)$ and $O(n^2)$ algorithms in seconds

2.4 Verification

3 Result Summary

The algorithm I created is clearly $O(n)$

References

[1] Zizhen Chen and David W Matula. "Bipartite Grid Partitioning of a Random Geometric Graph". In: *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE. 2017, pp. 163–169.

[2] Hichem Kenniche and Vlady Ravelomananana. "Random geometric graphs as model of wireless sensor networks". In: *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*. Vol. 4. IEEE. 2010, pp. 103–107.