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

Luke Wood

March 5, 2018

1 Executive 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]. 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.

1.1 Introduction and Summary

As a functional programming enthusiast, I first wished to use Elixir to solve this problem but quickly realized that the enforcement of immutability by the erlangVM would be problematic going forward. Due to the need for mutable data I decided to go with python for my implementation of part one. Despite choosing to use python, I really dislike the style most python programmers use and find it ambiguous and confusing in many cases. To counteract this, I employed functional programming patterns inside my python implementation such as decoupling all of my code as much as possible, explicitly importing modules for the function they are used in, and avoiding branching on conditionals whenever possible. I believe that these helped me create low bug code.

1.2 Programming Environment Description

I developed and tested this program on a 15 inch Macbook 2017 with an i7 processor and 16 GB of ram.

2 Reduction to Practice

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		

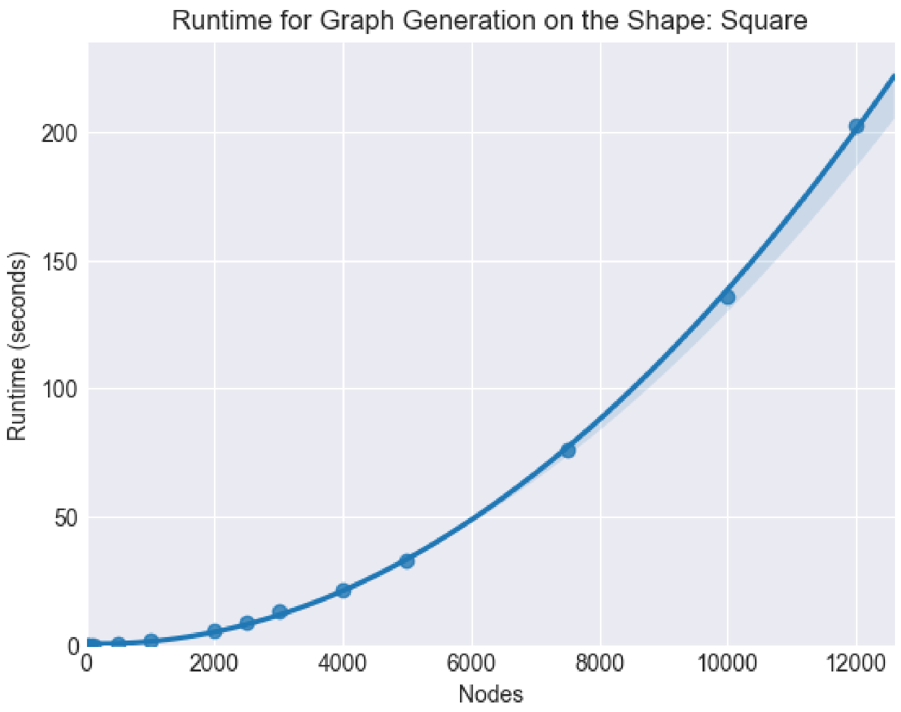

Figure 1: Data on the Runtime of the $O(n^2)$ Algorithm

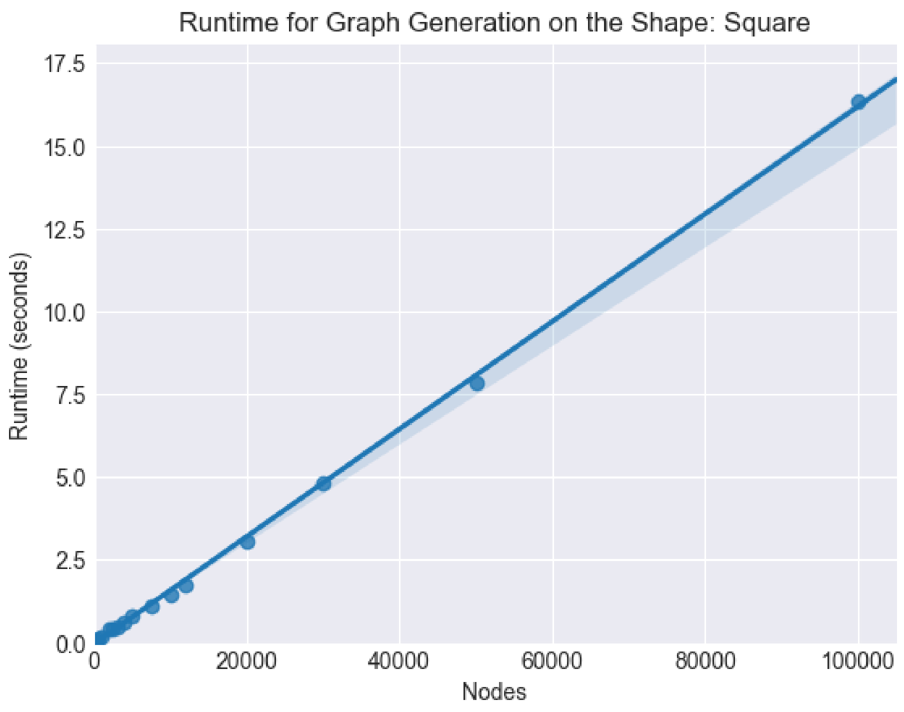

Figure 2: Data on the Runtime of the $O(n)$ Algorithm

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.