

Backbone Determination in a Wireless Sensor Network

Jake Carlson

February 18, 2018

Abstract

A report on implementing algorithms to partition a random geometric graph into bipartite subgraphs. Three different graph geometries are explored: unit square, unit disk, and unit sphere. Nodes are uniformly distributed in the geometry. Then the edges are determined and ...

Contents

1	Executive Summary	3
1.1	Introduction	3
1.2	Environment Description	3
2	Reduction to Practice	3
2.1	Data Structure Design	3
2.2	Algorithm Descriptions	3
2.2.1	Node Placement	3
2.2.2	Edge Determination	3
2.3	Algorithm Engineering	4
2.4	Verification	4
3	Results Summary	4
4	Appendix	6

1 Executive Summary

1.1 Introduction

1.2 Environment Description

The data structures and topologies for this simulation are implemented in Python2.7. The graphics are done using Processing.py. All development and benchmarking has been done on a 2014 MacBook Pro with a 3 GHz Intel Core i7 processor and 16 GB of DDR3 RAM.

2 Reduction to Practice

2.1 Data Structure Design

The primary data structure used for this project is an adjacency list. However, to allow for constant time lookup of edges of a node, I used a Python dictionary where the keys are nodes and the values are a list of adjacent nodes.

2.2 Algorithm Descriptions

2.2.1 Node Placement

A different node placement algorithm is required for each of the geometries. For the square, the coordinates for each node are generated as two random numbers taken from a uniform distribution on the range 0 - 1. All of these points are guaranteed to be in the unit square.

For the disk, a similar method is used. The coordinates for nodes are randomly sampled from a uniform distribution; however, if a node has a distance from the center of the disk greater than the radius of 1, the coordinates are resampled.

For the sphere a different method must be used so that all of the nodes are placed on the surface of the sphere and the volume is vacant. For this geometry, I used the following equations:

$$x = \sqrt{1 - u^2} \cos \theta \quad (1)$$

$$y = \sqrt{1 - u^2} \sin \theta \quad (2)$$

$$z = u \quad (3)$$

where $\theta \in [0, 2\pi]$ and $u \in [-1, 1]$. This is guaranteed to uniformly distribute nodes on the surface area of the sphere [1].

2.2.2 Edge Determination

I implemented multiple methods for finding the edges in the graph. The brute force method iterates over every node, and for each node checks all other nodes to see if they are close enough to form an edge. The brute force method is $\Theta(n^2)$ where $n = |V|$.

The second method I implemented to find the edges is the sweep method. This method first sorts the nodes along the x-axis. Then, for any node, we only need to search left and right until the distance along the x-axis is greater than the connection radius for the nodes. This dramatically reduces the search space. The sweep method is $O(n \lg(n) + 2rn^2)$ where $n = |V|$ and r is the connection radius. The $n \lg(n)$ portion is for the sorting and the $2rn^2$ portion is for measuring the distance between nodes in a sweep step.

The final method I implemented to find edges is the cell method. This method places the nodes into cells of area $r \times r$ based on their position in the topology. When the edge detection runs, each node needs to be visited once, but only the cell the node populates and the neighboring cells need to be searched for connections.

2.3 Algorithm Engineering

2.4 Verification

3 Results Summary

References

- [1] Weisstein, Eric W., Wolfram MathWorld *Sphere Point Picking*
<http://mathworld.wolfram.com/SpherePointPicking.html>

