Generating Connected Waxman Graphs

April 17, 2018

1 Connected Graph Generation (CGG) algorithm

Here we give an explanation of the generation of connected Waxman networks, a type of Spatially Embedded Random Network (SERN), with the Connected Graph Generation (CGG) algorithm.

Initially we require a random network **G** generated from the distribution of interest. Here we are interested in the Waxman network where each node is connected with probability

$$p_e = qe^{-sd}, \tag{1}$$

see below for more details.

As graph constructions can hold position information in various ways, the input to the CGG algorithm requires two position vectors (lists in python), **position_x** and **position_y**, for the x and corresponding for y coordinates.

We use the Waxman *distance deterrence function* in (1) for the **probability** input in the CGG algorithm. Other distance deterrence functions can be used to generate different SERNs.

The **number_of_iterations** is simply the number of steps desired in the Metropolis-Hastings process. This value should be decided on a case by case basis depending on the size and type of graph. For Waxman networks of $n = 100$, one million iterations is very conservative.

The final input in the CGG algorithm is the **record_step**. The algorithm is designed to save the average degree at certain steps in the process. This allows for exploration of the convergence properties.

The CGG algorithm can easily be extended to save different statistics of the graph through the process; as well as using a different distance metric to generate a different type of SERN.

2 Waxman Networks

We use the Waxman parametrisation in (1) where $q \in (0, 1]$, $s \geq 0$, and d_{ij} is the Euclidean distance. The parameter s controls the extent to which spatial structure is incorporated into the graph. Note that when $s = 0$ we recover the Erdös-Rényi random graph.

The q value is the thinning of edges in the graph. Note that the (1) differs from much of the literature on Waxman graphs. We follow [1] and use this as unfortunately, the parameters (α, β) used traditionally have become confused by frequent reversal. Another benefit of this parametrisation is the ease of tuning desired parameters of the graph, most notable the average node degree.

The function `waxman_generator` takes the desired `s` value and average degree (`z` and uses the methods described in [1] to transform these values into the required inputs for the NetworkX package. If the parameters α and β are desired inputs, this step can be skipped and the inbuilt generator used.

3 References

References

[1] M Roughan, J Tuke, and E Parsonage. Estimating the parameters of the Waxman random graph, 2015. arXiv preprint: 1506.07974.