Random Graph Models

Adam Gibbs, Quin McElhiney, Raymond Saitoti

5 May 2020

Introduction

The question of how one can generate synthetic, realistic, and time-evolving graphs bares great relevance on such enterprises as graph mining, finding abnormalities in networks, conducting extrapolations, simulating possible what-if scenarios, and studying the laws that a graph generator should obey in order for its generations to be realistic.

The earliest stochastic generative model for complex networks was the random graph model introduced by P. Erdos and A. Renyi in "On The Evolution of Random Graphs"(1960), in which each node pair, $(i, j) \in V(G)$ has an equivalent, independent linking probability, p. This generative model, while permitting for extensive mathematical theory, produces graphs which fail to match real-world networks, especially in their lack of heavy-tailed degree distributions. The majority of the models which attempted to mitigate the noted short-comings of the ER-Model advanced some variation of *Preferential Attachment* (See Albert and Barabasi in "Emergence of Scaling in Random Networks" and "Statistical Mechanics of Complex Networks") in which new nodes are added to an existing graph sequentially, and are connected preferentially to nodes with higher degrees. These models, while producing power-law tails and low diameter networks, failed to produce graphs whose diameter grows slowly with $n := |V|$ (also called the "Shrinking Diameter Property").

In an attempt to create synthetic graphs that exhibited the observed properties of natural graphs, many models after the ER Model and the Barabasi-Albert Preferential Attachment Model were formalized and studied. This has given rise to the sub-field of random graph generation within network science and within the sub-field of random graph generation there are many different approaches. The two main groups will be explored here and those are static and time-evolving networks. Static models were the first models to be developed and began with the ER Model leading into many other such as the Small-World Watts Strogatz model, Stochastic Block Models, Random Dot Product Model, Exponential Family Models, and the Kronecker Product Model. From here, there came a desire to model the evolution of these models over time, and the creation of evolving models began with the Barabasi-Albert Model and led to the Fitness Model and Various Local-World Models. Each model presented in

literature comes with its own strengths and weaknesses and real-world applications and we will discuss those further in the next sections.

Static Models

Static models beginning with the ER model were designed to emulate real world networks and their characteristics to help prove properties or see how networks would behave in different simulated scenarios. The main characteristic of a static model is that it is not time dependent, it has a fixed graph size that it given when the model is used to create a graph and that does not change. Many of these models artificially replicated certain types of networks (i.e. online social networks) well and allowed network scientists to perform simulations or tests they needed. However, most proposed generators focus on a small number of static patterns and neglect others, leading to models which either fail to generate graphs capable of matching the evolving complexity and characteristic richness of real networks or are too complex to be mathematically tractable. These issues are addressed in Leskovec et al. "Realistic, Mathematically Tractable Graph Generation and Evolution, Using Kronecker Multiplication", wherein they introduce The Kronecker Graph Generator and demonstrate its superiority to competing models. We will first discuss two random graph models that improve on the original ER and Barabasi-Albert Models but still carry significant limitations in the generation of natural, synthetic data. Then we will discuss the Kronecker Graph Model which is commonly considered the best model for generating natural synthetic graphs.

Latent Space Models and the Random Dot Product Model

As the limitations of the ER and Barabasi-Albert Models led to newer models being introduced, the concept of latent space models began to be discussed. They were first proposed in 2002 by Hoff, Raftery, and Handcock [2] and were developed in response to ideas of "social space" being discussed and its notion that actors are more likely to know other actors who are connected to their neighbors in a graph. To model this, latent models assign random positions in space and the probability two actors are connected is related to their distance from each other in space. A common implementation of this is comparing the distance between two points in space represented by vectors of an arbitrary dimension. Latent space concepts became popular when geometry was used in attempt to mimic characteristics of networks not found in other graph models such as directed cycles and non-uniform associativity [1]. Another advantage to these models is that they also provide simpler ways to visualize graphs.

One of the more successful models that extends off of other latent models was the Random Dot Product Model (RDPM) proposed by Kraetzl, et al [3]. In the RDPM, nodes are characterized by random vectors in d-dimensional space, $\mathbb{R}^d$. The vectors are formed by drawing `d` real numbers from a selected distribution. The probability that a node is connected to another node is given by

the dot product of the two vectors and then a transformation that maps the dot product to the interval $\mathbb{R}[0,1]$ [4]. The idea behind using the dot product as the probability two nodes are connected is that the dot product encodes two forms of "similarity." Both the angle between two vectors and the magnitude of the vectors factor into the dot product, so vectors of similar magnitude and direction are therefore more alike, and have a higher probability of being connected. So when creating a random graph, the RDPM creates random nodes first, and then connects them based on their similarity [3]. However, the RDPM has limitations. The most obvious being that it is relatively dense with its edge density being on the order of $\Omega(N^2)$, whereas, most real networks are sparse. This limitation, however, simply narrows the use of the RDPM. It is used most commonly with social networks as social networks become more and more dense, ultimately making the density of the RDPM desirable. Other dense networks including some gene and protein networks and networks that model the structure of the brain, also use RDPMs to generate synthetic data. So although the RDPM works well with dense networks, its narrow use leaves lots of work left to be done in creating a more adaptive model that accurately models a wider breadth of natural graphs.

Small world "Watts Strogatz" graphs

Small world graphs are a hybrid between random graphs and regular graphs. Similar to dot product models, small world graphs were introduced as an attempt at exploring models exhibiting multiple tendencies. The original paper exploring the dynamics of these small world networks credits the development for this model to the need to explore networks that can be characterized as both normal and random. Just like random dot product models, these models try to explain the small world phenomenon in networks such as biological, technological and social networks (Watts and Strogatz, 1998). The process of modelling these networks involves the rewiring of normal networks, with the aim of modelling the properties of hybrid networks. The coined phrase "six degrees of separation" emanates from the tendency of these networks to exhibit and model properties traditionally observed within the mentioned networks.

Kronecker

There are a number of challenges posed by working with natural graphs. These challenges include: that *the difficulty of obtaining natural data-sets is high*; *existing data-sets for natural graphs, and of sufficient size, are few*; *synthetic graphs lack certain characteristics found in natural graphs.* These "characteristics of natural graphs" include the property of having a degree distribution following a power law probability distribution and the property of self-similarity, or of large scale connections between parts of the graph reflecting the constituent subgraphs. An accurate generative model (where accuracy is measured with respect to natural graphs) will preserve these properties. Thus enters the stochastic kronecker graph model.

At the essence of The Mathematical Formulation for the Kronecker generation model is the so called *Kronecker Product*: a 2-ary matrix operation in which one operand is "nested" or copied inside the other. The application of this operation to the adjacency matrices of graphs is the novel step in the Kronecker Model. Using the Kronecker Product on a graph and its copy can easily produce a self-similar graph (as does taking the similarly defined *Kronecker Power*). This simple loop: of repeatedly performing the Kronecker operation on an initiator graph, readily — and deterministically — produces accurate synthetic networks. While graphs generated thusly have a variety of desired properties, their discrete nature produces unwanted staircase effects in the degree and eigenvalue distributions (mostly because of the high multiplicity of individual values). To generate *random*, natural graphs, we start with an adjacency matrix, P, for which $P_{ij} \in [0, 1]$. This gives us the *stochastic* Kronecker Model.

The properties of stochastic Kronecker Graphs, discussed extensively in [13] and Nazim, et al. in "Properties of stochastic Kronecker Graphs", include the following: I) Kronecker Graphs have multinomial degree distributions for in- and out-degrees; II) The Kronecker Graph G_k has a multinomial degree distributions for its eigenvalues; III) Kronecker graphs follow the Densification Power Law (DPL) with a densification exponent: $a = log(m)/log(n)$.

Evolving Models

After creating many static models that emulated the properties of real graphs, it became valuable to understand how graphs can change over time. This is where the evolving models began. The first evolving model was the Barabasi-Albert Preferential Attachment Model that begins with a small initial graph and at each time step t adds one node and links it to m existing nodes with a probability that the new node links to an existing node that is proportional to the existing nodes degree. This model promotes the idea that the rich get richer since higher degree nodes get more links and then the probability a node links to them increases further. Limitations to this approach became apparent in the study of real networks such as the internet. In the case of actors on the WWW, nodes like Google and Facebook entered the network later and ended up with a higher degree than the highest degree nodes when they intially entered the network. In the Barabasi-Albert Model, this would not have been possible. This is when the idea of fitness was introduced into evolving models.

Fitness and Evolving Models

The idea of fitness is that each node has a fitness attribute that determines its ability to gain links to other nodes. A node of high fitness has a higher probability of being linked by other nodes. The first model to implement this idea was the Bianconi-Barabsi Fitness Model. The Fitness Model works very similarly to the Barabasi-Albert Model as it uses preferential attachment to link

all incoming nodes to m existing nodes at each time step t. However, in the Fitness model the probability that an existing node get linked to is proportional to the product of the existing node's degree and fitness. By adding links in this way, a node has a greater chance of achieving a high degree if it enters the network early or it has a high fitness, similar to what we see in real networks. The Barabasi-Bianconi Model laid the framework for more accurate modeling of how networks evolve.

How Models Simulate Network Evolution

After the Fitness Model introduced the idea of fitness, other dynamics aspects were added to evolving network models to more accurately model the evolution of networks. Some of these include:

1. **Node Deletion.** In real networks, nodes leave the network as well as get added. By implementing a form a node deletion to a model, it more realistically models how a network changes over time. A common way of including this in a model is to delete a random node, and all its adjacent edges, at a time step t with some probability p rather than adding a node.

2. **Internal Links.** In the first preferential attachment models, the only new edges added to the graph were between incoming nodes and m existing nodes. However, in real networks two existing nodes can form a link between them. So evolving models with internal link implementations have a way to randomly add edges between existing nodes at certain times.

3. **Accelerated Growth.** In real networks, the evolution of a network may not be linear. With accelerated growth at each time step t, an incoming node links to m existing nodes where $m = m_0 t^{\theta}$ and θ is a parameter to the model such that the number of nodes an incoming node links to changes over time. This results in the size of the graph to change non-linearly and the average degree becomes dependent on time.

4. **Aging.** In real networks the age of a node usually has some influence on its fitness. So in an evolving network model that models the effect of aging, the probability an incoming node links to an existing node is related to the time that a node has been in the graph. The model can either give preference to younger or older nodes and can adjust how strong that preference is when performing the preferential attachment.

5. **Local-World Attachment.** In a real network when a node is added and that node has to decide what node to link to, it probably only has knowledge of a small portion of the network. So, the new node will link to one of the existing node that is known to them. For example, on social media someone will follow, link to, someone that they know and the set of people they know is most likely a subset of all the nodes in the network that is much smaller than the set of all nodes in the network. So in a local-world model, at each time step t, M existing nodes are chosen to form the

local world and the incoming node selects m nodes from the local-world to attach to.

Benefits of Evolving Models vs Static Models

Evolving networks by definition are networks that change as a function of time. They can be conceived as sequences of static graphs which change according to a dependent variable. They can be used to more accurately study real world networks which typically evolve as a function of time through sequential adding or subtracting from nodes or edges. Actor-level network measures like network centrality change as a function of time, importance and influence of individuals grow or decline depending on processes, and events occur in networks during intervals of time. Such problems as graph statistics computation, link prediction, community detection, and visualization are easier to consider when studied on evolving networks.

Concluding Remarks

Here we explored the two main categories of random models, static and evolving, including a select few specific random models from the literature. These models range from the simplest models with the ER and BA models to much more complex models like the Kronecker Product Model. In our world where there are millions of different types of networks, there are similarly many different types of random models to produce artificial graphs that fit the network properties you are exploring. We found that almost all models have some significant limitations whether it be a common network property that it does not replicate or that they can only replicate a very specific type of network. But there are some networks, like the Kronecker Product Model that have the ability to fit almost all types of networks. It was in the limitations that we discovered the lack of a time dependence in many models and led us to explore evolving models. Evolving models were interesting as many of the models were algorithms for adding or removing nodes over time beginning with an initial graph. So, many of the static models we initially explored could be extended with one or more evolving model algorithms. And the ways that which different models try to accurately model the evolution to mimic the many different ways that networks evolve are so diverse that they can be intertwined to make larger, more beautifully complex models. We looked into about 5 ways that network evolution is modeled with node deletion, internal link addition, accelerated growth, aging, and local-world attachment, but there are more as well. There are community based structures where communities can grow, shrink, merge, and split, or a combination of those. There is so much value is understanding the evolution of networks and it is so complex that there is so much information and work to still be done making it a fun topic to explore.

Given more time, we would've continued exploring evolving networks and

seeing how different initial graphs evolved after running an network evolution algorithm on it. For example, in the local-world node-deletion model, how does the final graph after t time steps change when the initial graph is a Kronecker graph rather than an ER graph or a Random Dot Product graph. The extension of static graphs with network evolution models is an interesting one with many real world applications as well. So, also if we had more time we would explore applications of evolving network models and how different phenomena that we observe in the real world could be modeled by different network evolution models. In a sense, being able to predict–or know with some certainty–what a network will look like after a set period of time is like knowing the future and as we said before it has great value, so we would have continued exploring that in the future. In the end, this project opened our eyes to the academic world of random model generation and showed us its broad applications from purely theoretically to the purely applicable–such as using the Kronecker model as part of a super-computing standard test.

Implementations

Of the models referenced in this discussion, we have written code implementations of the following models on GitHub:

1. Random Dot Product Model
    - Implementation: Random Dot Product Model.ipynb
    - Description: RDPM_README.txt
2. Watts-Strogatz Small-World Model
    - Implementation: SmallWorld.py
    - Description: README.md
3. Kronecker Product Model
    - Implementation: KroneckerGraphGenerator.py
    - Description: README.txt
4. Bianconi-Barabasi Fitness Model
    - Implementation: Bianconi - Barabasi Model.ipynb
    - Description: BB_README.txt
5. Accelerated Growth Extension of the Barabasi-Albert Model
    - Implementation: ExtendedBA.py
    - Description: README.txt
6. Local-World Node Deletion Model
    - Implementation: Local World Node Deletion Model.ipynb
    - Description: LWNDM_README.txt
7. Local-World Aging Model
    - Implementation: AgingLocalWorld.py
    - Description:ALW_README.txt

References

[1] Flaxman, A.D., Frieze, A.M., Vera, J.: A geometric preferential attachment model of networks. Internet Math. 3(2) (2006) 187–205.

[2] Hoff, P. D., Raftery, A. E. and Handcock, M. S. (2002). Latent space approaches to social network analysis. Journal of the American Statistical Association 97 1090–1098.

[3] Kraetzl, M., Nickel, C., Scheinerman, E.R.: Random dot product graphs: A model for social netowrks. Preliminary Manuscript (2005)

[4] S. J. Young and E. R. Scheinerman. Random dot product graph models for social networks. In WAW '07: Proceedings of the 5th Workshop On Algorithms And Models For The Web-Graph, pages 138–149, 2007.

[5] Duncan J. Watts and Steven H. Strogatz, Collective dynamics of small-world networks, Nature, 393, pp. 440–442, 1998.

[6] Bollobás Béla (2004). Random graphs. Cambridge: Cambridge Univ. Pr.

[7] Guare, J. (2010). "Six Degrees of Separation". BLOOMSBURY Publishing PLC.

[8] R. Albert and A.-L. Barabasi. Emergence of scaling in random networks. Science, pages 509–512, 1999.

[9] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. Reviews of Modern Physics, 2002.

[10] J. M. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The web as a graph: Measurements, models and methods. In Proceedings of the International Conference on Combinatorics and Computing, 1999.

[11] S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Extracting large-scaleknowledge bases from the web. VLDB, pages 639–650, 1999.

[12] A. N. Langville and W. J. Stewart. The Kronecker product and stochastic automata

[13] Leskovec, J., Chakrabarti, D., Kleinberg, J., and Faloutsos, C. 2005. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In Knowledge Discovery in Databases (PKDD'05), A. Jorge, L. Torgo, P. Brazdil, R. Camacho, and J. Gama, Eds., Lecture Notes in Computer Science, vol. 3721. Springer, 133–145.

[14] Gu, Yuying, and Jitao Sun. "A local-world node deleting evolving network model." Physics Letters A 372, no. 25 (2008): 4564-4568.

[15] Wen, Guanghui, Zhisheng Duan, Guanrong Chen, and Xianmin Geng. "A weighted local-world evolving network model with aging nodes." Physica A: Statistical Mechanics and its Applications 390, no. 21-22 (2011): 4012-4026.

[16] Li, Xiang, and Guanrong Chen. "A local-world evolving network model." Physica A: Statistical Mechanics and its Applications 328, no. 1-2 (2003): 274-286.

[17] Bianconi, Ginestra, and A-L. Barabási. "Competition and multiscaling in evolving networks." EPL (Europhysics Letters) 54, no. 4 (2001): 436.