Feature Constrained Graph Generation with a Modified Multi-Kernel Kronecker Model

Federico Reyes Gomez [1] **Vamsi Saladi** [1] **Markie Wagner** [1]

Abstract

Graph generation algorithms are a cornerstone of research on graphs as random graphs can serve as useful points of comparison, anonymized versions of real-world graphs, or hypothetical synthetic datasets that can be used to test novel algorithms. A key factor of these graph generation algorithms is that they output realistic graphs. Realistic in this context typically refers to requirement that they exhibit some of the same features as a real target graph, such as small diameters in small-world graphs, certain motif distributions, or clustering coefficients. Thus the problem of creating a random graph that is constrained to certain features is very important. Currently, most methods take a real graph as a reference and then optimize a graph generation algorithm to output a random graph that has similar features to this one specific real graph. We see a need to develop a more general graph generation algorithm that will fit certain features but one that does not require a reference graph. This will allow for more flexibility in generating realistic random graphs that don't exactly match a certain existing graph.

1. Context

Our goal for this project is to design a graph generation algorithm that outputs a graph that fits the specified input features. We envision a general algorithmic framework that, given graph features such as the clustering coefficient, degree distribution, motif counts, and/or the spectrum of its adjacency matrix and laplacian matrix, can output a random graph of size n that has these features. Possible applications include generating random graphs based on real-world graphs for privacy and anonymization reasons, or for generating graphs to fit hypothetical situations that researchers may want to study.

This generation algorithm is based on the Kronecker Graph Generation algorithm [1] proposed by Leskovec, et al. This algorithm takes a single initiator kernel and repeatedly applies the Kronecker matrix product to generate larger graphs. We propose first determining a set of base kernels to choose from and, instead of multiplying by the same one each time, multiply by different kernels at different levels. We propose this as an alternative (or addition to) stochastic Kronecker graph generation in order to add randomness and variability to the output graphs.

In order to test our algorithm, we plan to first use synthetic datasets, probably created using the Kronecker graph model. We'll then try to fit various real graphs available on the SNAP group's website. First, we'll try a biological network since they are small and allow us to test our idea very quickly. Then, We'll try a social networks which have a great use sociologically and then, time permitting, attempt to use our algorithm on other graphs. In order to make this a tractable project, we'll focus on smaller real-world graphs.

Table 1. Description of the Datasets with High Level Features

NAME	NODES	EDGES
DISEASOME	516	1188
SC-TS	636	3959
SC-HT	2084	63027

2. Datasets

To start testing out various ideas, we first chose a dataset. We focused on biological datasets because of the relatively small size of the graphs in this domain. We used the Biological Network Repository source to find small biological networks. We ended deciding on the following three graphical models:

2.1. Bio-Diseasome Network

This is a network that details out the human disease network the and set of common genes that are known to be associated with each of these networks. As we can see from Table 1. this is a network that is relatively small and only has 516 nodes and 1188 edges. We can visualize the Bio-Diseasome Network below:

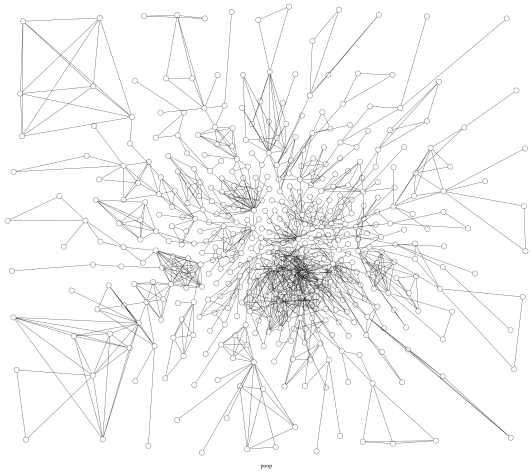

Figure 1. Bio-Diseasome Network visualized

We have the following list of additional qualities that we know about this graph:

- Minimum Degree: 1
- Maximum Degree: 50
- Average Degree: 4
- Average Clustering Coefficient: 0.63583
- Density: 0.00894

2.2. Bio-SC-TS

The Bio-SC-TS network details out the interaction of nucleic acid and can be visualized here:

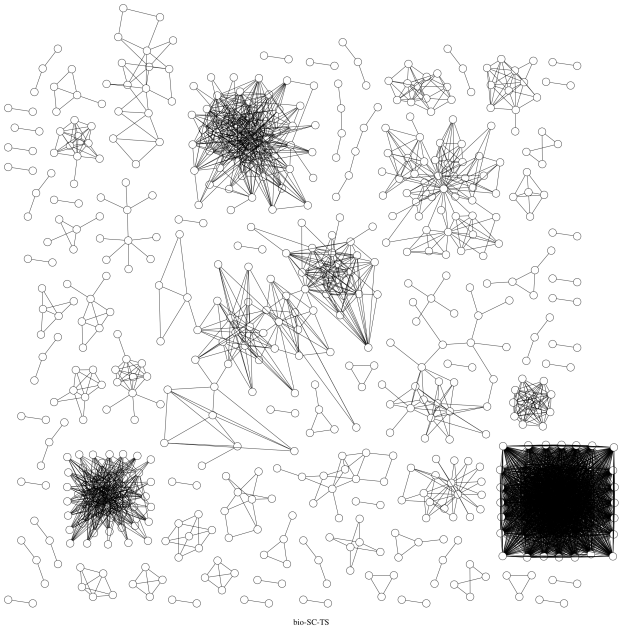

Figure 2. Bio-SC-TS Network visualized

This is a graph with 636 nodes and 3959 edges, and We can see that there are a lot of fragmented pieces to this network that are very connected. We have the following list of additional qualities that we know about this graph:

- Minimum Degree: 1
- Maximum Degree: 66
- Average Degree: 12
- Average Clustering Coefficient: 0.47124
- Density: 0.01961

2.3. Bio-SC-HT

This is a graph that deals with nucleic acid interactions with the predictions of RNAi phenotypes. The graph has 2084 nodes and 63,027 edges. This network is unfortunately too large to picture here, so we have just detailed out the other aspects of this network:

- Minimum Degree: 1
- Maximum Degree: 472
- Average Degree: 60
- Average Clustering Coefficient: 0.3491
- Density: 0.02904

3. Initial attempts

In order to make this project tractable, we first had to make some decisions to limit the scope of the project. First of all, we decided that coming up with a unique loss function would be too ambitious for a single-quarter project. Thus we decided to focus on the backtracking approach similar to [2] instead of a loss function, gradient-descent type of approach.

Instead, to measure graph similarity, we will use pre-exisiting, state of the art similarity metrics to evaluate our graph generation algorithm. Some of the similarity metrics we would like to consider are: Substructure Index-based Approximate Graph Alignment (SAGA) and Approximate Constrained Subgraph Matching as evaluation metrics.

Next we decided to first determine what are the best features to test such an algorithm.

One large part of our project depends on being able to generate initiator kernels that fit the desired features. In theory, we could then use these to generate a larger graph by taking repeated Kronecker products.

We initially tried to focus on using the spectrum of the Adjacency matrix as the feature to model. We took our single test graph, the bio-diseasome graph, and found the spectrum and related eigenvectors of it. Since we want to generate smaller kernels (of size 3 for our first attempt) then we took the 3 largest eigenvalues of the target graph,

generated a random basis for $\mathbb{R}^3$. We then generated a new adjacency matrix $A' = \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T + \lambda_3 v_3 v_3^T$. We then divided by the max to normalize and generated a new graph by taking all values greater than 0.5. Finally, we constrained the generated graph to have the same number of edges as the target graph. After this was all done, we plotted the graph and, to no surprise, we didn't get very good results.

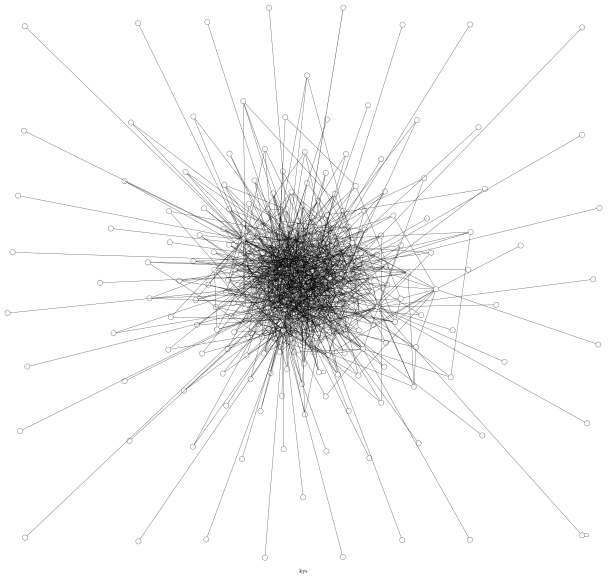

Figure 3. First Attempt at a Generated Graph

This looks very much like a random graph and visually not at all like our target graph. We also did some quantitative checks, first with the clustering coefficient. The clustering coefficient of the target graph was 0.63583 and the clustering coefficient of our generated graph was around 0.09.

From this experiment, we realized that the spectra of the Adjacency matrix alone is not expressive enough to maintain structure of a graph. Note that in this case we generated a graph of the same size of the target node to test our initial assumption that the spectra would retain features of the graph, but in our actual algorithm the intent of this would be to generate the small 3 or 4×4 initiator kernels. Nevertheless, we see that this approach doesn't work.

4. New Direction

Since we realize that generating initiator matrices that are constrained to some set of features is too difficult of a task in and of itself, we plan to pivot to a new direction. We plan to keep our general algorithmic framework of using a set of initiator matrices and iteratively generating larger and larger graphs, but we intend to use motifs instead. The reason for this is that

1. *Motifs are more concrete and easy to enumerate for small graphs*

   If we want to generate initiator matrices that correspond to different motifs, it's easy to enumerate over all 3 and 4 size graphs. Additionally, the isomorphism of these graphs allows us some degree of freedom that allows us to add a stochastic element to the algorithm

2. *Motifs are more easily interpretable than other complex features*

   One of the initial goals of the graph was to be able to find a graph generation algorithm that doesn't depend on a target graph. Ideally, a researcher should be able to specify features of a theoretical graph and then be able to generate a concrete version of it. Motifs and motif counts are both important and easily interpretable features of graphs that someone would have reasonable guesses about. Compare this to a spectrum or the second principal component of the Laplacian matrix, very expressive features, but not very easily interpretable or a priori predictable.

We realize that we have drastically changed the nature of our features and thus we'll need to start from scratch again, but we believe this is a much more feasible direction to go in than using spectral features, which would require a lot more rigorous mathematical analysis than is feasible for this project.

5. Literature Review

5.1. Graph Generation

Kronecker graphs were proposed by Leskovec et al. as a way to detail a mathematical approach to generating graphs given certain features (Leskovec et al., 2009). Specifically, given some structural property and a subgraph to emulate, the paper details an algorithm for generating larger graphs that emulate the substructure and maintain self-similarity. The paper does this by detailing a mathematical concept (Kronecker product) that helps maintain a self-similar structure while expanding the size of the adjacency matrix. By generating various different graphs with this algorithm, the paper shows that graphs generated by this method are effective and have similar properties to graphs found in the real world (like heavy tails for the in-degree and out-degree distribution, heavy tails for the eigenvalues and eigenvectors, small diameters, and densification and shrinking diameters over time. Additionally, the paper also provides an algorithm (KronFit) that can be used for fitting the Kronecker graph generation model to large real networks and does so in linear time rather than the naive exponential time.

While the paper presents a very efficient way of finding parameters (an initiator matrix) to fit a real world graph, it still

requires a reference. Say you're a scientist studying global economic flow and would like to study a proposed trade deal under a hypothetical condition with higher interconnectivity, you would need to generate that graph somehow in order to study it. Additionally, while not a limitation per se, it would be interesting to study graphs that are generated as a result of Kronecker multiplications between different matrices, instead of recursively multiplying the same one over and over again. The stochastic Kronecker generation method was proposed to vary the graphs generated, but we plan to add even more variance and get more expressive output graphs by using multiple kernels. We plan on using a modified version of the Kronecker model as our generation method. The Kronecker model depends on using some initiator structure that can be replicated and duplicated to generate larger graphs. However, there is usually more than one initiator substructure that might fit the necessary constraints and features. In that case, we want to be able to alternate and iterate through various possible kernels, and see if using different kernels at each step can help us get us large graphs that can also satisfy our constraints, and see if they are closer to the ground-truth graphs. Additionally, the paper also discusses the possibility of going from a ground-truth graph back to an initiator kernel and then to a random graph; this is a process that we can modify (since we want to go more strictly from features to the random graph), and thus helps provide some insight into the MLE approach to graph generation and feature extraction.

5.2. Prescribed Feature Constraints

In a paper by Ying and Wu, four main features are used to constrain generated graphs: the spectrum of the adjacency matrix, the second eigenvalue of the Laplacian matrix, the harmonic mean of the shortest distance, and the transitivity measure of a graph (Ying & Wu, 2009). They chose these features because of their expressiveness and their relation to more traditional graph features such as the maximum degree, chromatic number, clique number, and extent of branching in a connected graph. They propose a generation algorithm based on edge switching: Given an input graph G to model, randomly switch edges if the resulting graph still satisfies the specified features. If not, revert the switch and choose a new random edge. They empirically show that this method is effective in preserving various key features that they analyze.

Ying's paper presents a novel way to fit a graph to certain features.KronFit implicitly fits a graph to certain features by using a Maximum Likelihood Estimation method, while the proposed switching algorithm fits the graph to specific user-specified features. This is closer to what we want, but this switching algorithm still requires a reference graph that serves as the input to the algorithm. Additionally, this paper is much less rigorous than the Kronecker paper, although that is something we can use to our advantage.

This paper presents a feature-constraining paradigm for graph generation that we'll also be targeting. Secondly, it presents four key features that we would ideally try to use in our algorithm, although we intend to focus on motif features moving forward. The paper also presents this backtracking optimization algorithm that, though less precise than a gradient descent optimization algorithm, may have better practical results with our problem. Finally, it prevents a good literature review of other graph-generation paradigms that we should take into account when developing our own algorithm.

5.3. Measuring Graph Similarity

An important factor in graph generation algorithms is in evaluating it and determining similarity of different graphs. A paper by Koutra et al. focuses on the idea of graph similarity and subgraph matching, and how we can develop algorithms for both to return metrics (Koutra et al., 2011). For example, the problem of graph similarity can be summarized as a problem of finding a one-to-one matching between the nodes of one graph and the nodes of the other (if they node sets are the same size), or at least matching nodes from one graph to another (if the node sets are different sizes). More specifically, the paper explores various approaches and key attributes in determining graph similarity such as edit distance, and graph isomorphisms and key feature extraction. The paper also goes through the idea of using belief propagation and other iterative measures like SimRank, which is a successful algorithm for evaluating self-similarity in graphs. Additionally, the paper explores the idea of subgraph matching, and how it can use Substructure Index-based Approximate Graph Alignment (SAGA), a measure developed so that there are fewer constraints. For example, SAGA can ignore node gaps, node mismatches and graph structural differences and does not require any constraints to be designed in advance. The paper also explores the idea of using tensor decomposition and PCA analysis to determine further subgraph matching.

We see that although the paper does in fact explore various features of graphs and their mathematical properties to evaluate a similarity metric, we see that it does not in fact take advantage of many other attributes. For example, there is no indication that motif counts factored into their metric at all. We also don't see any indication that the authors used any sort of community metrics or role metrics in matching nodes, but relied more on their mathematical spectra and eigenvalue properties. Finally, though there is a lot of exploration into subgraph matching, there is not a lot of exploration into the idea of subgraph permutations, or the idea that we can find some sort of mapping from one subgraph in one graph to a subgraph in the other. This is an idea that

has been explored as an analogue to node mappings, and is something to consider if we use their similarity measures for our application.

This paper will help us determine how to evaluate the similarity between our generated graphs, other generated graphs, and ground-truth graphs. Additionally, the paper has other similarity metrics like the eigenvalue metrics, which we can implement directly as a baseline and see if we can improve upon as we go further in our research and study of these graphs. Finally, the paper also details what are considered classical distance/similarity measures we can use a baseline like cosine distance, "Euclidean distance", Spearman's correlation, and Linear Correlation. These measurements and their respective values as determined by testing done in the paper will give us an industry standard to work from and improve upon.

Acknowledgements

We would like to thank Jure Leskovec for providing us the opportunity to do this project and for providing us the knowledge necessary to carry out the research. We would additionally like to thank the Biological Network Repository for providing the datasets that we used for our project.

References

Koutra, D., Parikh, A., Ramdas, A., and Xiang, J. Algorithms for graph similarity and subgraph matching. In *Proc. Ecol. Inference Conf*, volume 17, 2011.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Leskovec, J., Lang, K. J., Dasgupta, A., and Mahoney, M. W. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

Ying, X. and Wu, X. Graph generation with prescribed feature constraints. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pp. 966–977. SIAM, 2009.