Complex Networks deployment for Power Grid Analysis: A study on grid failures and blackout

Rohit Venkata Sai Dulam
Computer Science
University of Rochester
Rochester, NY 14627
`rdulam@ur.rochester.edu`

Sai Pranav Chittella
Computer Science
University of Rochester
Rochester, NY 14627
`schittel@ur.rochester.edu`

Meizhu Wang
Data Science
University of Rochester
Rochester, NY 14627
`mwang74@ur.rochester.edu`

Abstract

A power grid is an interconnected network consisting of generators and consumers of the power,transmitted along the power lines. A failure in this system due to human errors or a natural calamity can have a cascading effect creating blackouts in the connected regions. This can have devastating effects on the consumers and the industries located in that region. This project is about trying to find the nodes which are highly connected and then experiment on the effects the removal of such nodes might have and then testing different strategies to find another major node to power the blacked out region without overloading any nodes which might lead to cascading failures. [1] [2]

1 Introduction

In consideration of the US Power Grids, most of the major blackouts in power grids have been generally caused by an initial event (for instance, critical loads) that unchains a series of cascading failures or generally blackouts, with very severe consequences that compound the load on the remaining nodes, which eventually lead to massacre of power failure throughout the network. To cite a few of the reasons behind the cascade failure, high winds during storms and hurricanes can lead to faults and damage to overhead transmission and distribution lines, either by debris being blown against the lines or even a tower collapse in extremely high winds. Hence, the study of cascading failures [3] in power grids (both in power transmission grids, distributed generation) is currently a vibrant topic which is being profusely investigated through complex networks. [4]

The complex networks are of interest to us and also difficult to analyze due to its size and complexity of these situations and their different theoretical approaches. In our analysis we have focused on three Rs : Reliability, Resilience and Robustness. [5] According to the IEEE, Reliability can be termed as The probability of its satisfactory operation over the long run. Robustness can be viewed as the degree to which a network is able to withstand an unforeseen event without degradation in performance. Resilience, which is the most important aspect of this project studies the ability of a power system to recover quickly after a blackout or cascading failures.

Based on all the things talked above, we in our project are going to do the following things.

1. Find out the importance of a particular node in the network using Graph Convolution Networks [6] and Different Centrality Measures. The Centrality measures used in our case are Betweenness Centrality, PageRank [7] Centrality and Out Centrality.
2. Once we have the most important nodes, we'll be using Percolation theory to remove nodes from the network and see how the network is behaving. The behaviour might include the distribution of load on neighbouring nodes when a node is removed etc which can be categorized as Resilience analysis.

2 Related Work

2.1 Neural Networks on Graphs

Modifying Convolutional Neural Networks and Recurrent Neural Networks for the task of inference on graphs is a very tough task. There are many such specialized architectures of the above talked Neural networks that have been used on graphs. These approaches are ultimately multi-layer neural network approaches. The first main approach Defferrard et al. approximate smooth filters in the spectral domain using Chebyshev polynomials with free parameters that are learned in a neural network-like model [8].The network we use in our project is a modification of the above-mentioned approach [6]

2.2 Graph Embeddings

Graph Convolutional Neural Networks [6] are one class of Neural Networks that are used to obtain both node level and graph-level embeddings from a given graph. Although Adjacency matrix tells us about a graph, not all graphs are small enough to be represented as such. Some graphs might have a million nodes which would lead to a million by million adjacency matrix. This is practically intractable. In order to overcome this, a lot of research has been going into creating embeddings of both the vertices in a graph and the graph as a whole. Embeddings will give us a lot of information about the nodes like what set of nodes does a particular node in a graph belong to, how can the graph be partitioned, etc. With the advent of Machine Learning and Deep Learning, a handful of very good approaches for both types of embeddings have come up in recent times. Most of the ideas are based on Word2Vec [9] model. In short, Word2Vec [9] tries to embed words into 300-dimensional vector space. Deepwalk [10] uses Random walk approach to find the embedding of each node. Node2Vec [11], a variant of Word2Vec [9] for graphs, uses Random walk approach as well, and slightly different from Deepwalk [10]. SDNE [12] is different from the previous approaches. It is designed so that embeddings preserve the first and the second order proximity. The first-order proximity is the local pairwise similarity between nodes linked by edges. It characterizes the local network structure. Two nodes in the network are similar if they are connected with the edge. They use Autoencoders to find the embeddings. Each vertex or node is sent through an Encoder, which encodes the node in a latent space, and again decodes or reconstructs it back. The latent space is used as the embedding and also used in the loss function. Graph2Vec [13] is another approach that embeds the whole graph.

2.3 Percolation Theory

To study the percolation properties of a network [14], a fraction of nodes are randomly removed, and the behavior of the giant component is studied. There is more than one way in which vertices can be removed from a network. In the simplest case the nodes could be removed purely at random. An alternative approach is to removal scheme is to remove vertices according to their degree in some fashion. For instance, we could remove vertices in order of degree from highest to lowest.In this project we are working towards modifying the percolation theory algorithm which removes the nodes randomly, by calculating the betweenness centrality of the nodes and removing the nodes with the highest betweenness centrality.

Traditionally the percolation process is parameterized by a probability ϕ, which is probability that a vertex is present or functioning in the network. In the parlance of percolation theory, one says that the functional vertices are occupied and ϕ is called the occupation probability. Thus $\phi = 1$ indicates that all vertices in the network are occupied (i.e., no vertices have been removed) and $\phi = 0$ indicates that no vertices are occupied (i.e., all of them have been removed)

3 Data Analysis and Modeling

3.1 Data set Overview

We have taken two different data sets for this projects, as the approaches that we are interested in, cannot be done entirely on the same data set due to the properties such directed network, degree distribution,etc.

3.1.1 Dataset1

The first data set we have taken is the "US Power Grid" which is obtained from the GridKit [15]. Broadly it consists 3284 Distributors, 3790 Generators, 9093 Transmitters. The data has been pre-processed before it is analysed,Every edge connection is uniquely identified with an attribute "l_id" , vertices id's v_id 1 and v_id2 with a link 'li_id' between them. Voltage transmitted through that edge is identified through the 'voltage' attribute, 'length' and the 'cables' can be understood.

The type can be categorized as transmitters/distributors/generators.

Generators do not have any incoming node, whereas distributors have one incoming node and many transmitting node. The rest are categorized as transmitters.

The attributes 'lon' and 'lat' are the location co-ordinates respectively. Here, voltages column is very important as when the nodes are not in the circuit, they do not have any voltage, except the Generators. There are 5 generators from the analysis we have done, which are the most import nodes in our graph.

3.1.2 Dataset2

We have taken the data-set of US Electricity department in this study which only comprises of the Western United States, with N=4941 nodes and K=6594 edges. The electric power grid is represented as an undirected graph, in which the N nodes are the substations (generators, distribution, substations or transformers) and the K edges are the transmission lines.In our analysis and pre-processing of this data set, we have assumed that all the nodes are considered same(no disparity between the generators,transmitters, transformers) and the edges are non weighted. To each edge between nodes i and j is associated a number eij in the range [0; 1] measuring how efficiently nodes i and j communicate through the direct connection. [16] [17]

3.2 Data Visualization

3.2.1 Degree Distributions.

4 Graph Embedding and Models

But why embeddings? How is it useful for this project? As we are trying to find failures in a power grid, it is essential to know what the most important nodes are, and how the failure of these nodes would affect the entire network. Where would blackouts be caused if such nodes are removed and how much load would fall on the neighboring nodes to sustain such failures? In order to do that, we have multiple centrality measures to get our job done. The centrality measures used in our project are Betweenness centrality, out-degree centrality, PageRank [7] centrality, etc. Since this a research and learning project, weve also used GCN [6] to understand how such networks work. With the embeddings from the GCN [6], well be trying to match them with the outputs of the centrality measures to see how GCN [6] performs when compared to the ground truth. This is a purely experimental process and might not lead to the best results due to some reasons. One, a GCN [6] requires two inputs to be given as seen from the basic network architecture, which the adjacency matrix or anything else that tells about the network structure and features for each node. Unfortunately, in the datasets were using for the sake of the project, there are no features for any of the nodes, except for the fact that any one of the modes might be a generator, distributor or a

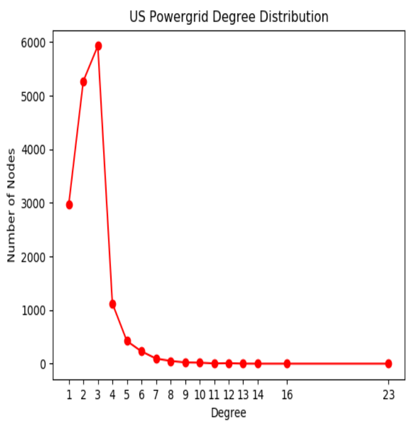

Figure 1: This figure presents the degree distribution of the first dataset.

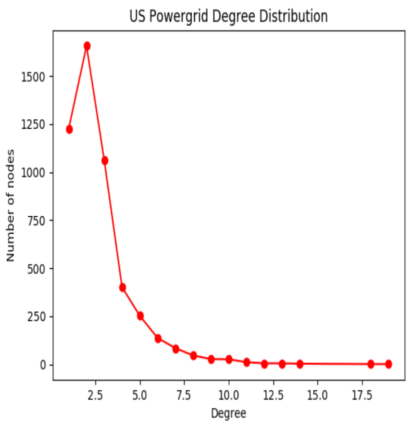

Figure 2: This figure presents the degree distribution of the second dataset.

transmitter. This analysis is also present only in the first dataset which is the Power grid network for the whole US. The other dataset does not have this information as well. Since this is all experimental, Weve tried to use the degree of the node to be its feature, which might lead to undesirable results.

4.1 Graph Convolutional Neural Network

The GCN [6] is used on a graph

$$G = (V, E) \tag{1}$$

which takes as input:

1. A feature description ***Xi*** which takes as input: for every node ***i***; summarized in a ***N by D*** feature matrix ***X*** (***N***: number of nodes, ***D*** : number of input features)

2. A representative description of the graph structure in matrix form; typically in the form of an adjacency matrix ***A*** (or some function thereof)

and produces a node-level output ***Z*** (an ***N*** by ***F*** feature matrix, where ***F*** is the number of output features per node).

Every neural network layer can then be written as a non-linear function

$$H^{(l+1)} = f(H^{(l)}, A) \tag{2}$$

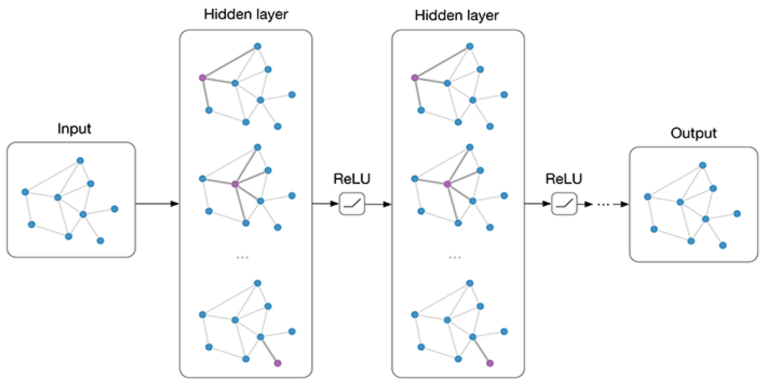

Figure 3: GCN [6]

where,

$$H^{(0)} = X \tag{3}$$

and

$$H^{(L)} = Z \tag{4}$$

where L being the number of layers of the GCN [6].

5 Achieved Results

We have collected the data and pre-processed both the data sets. Also, we have built the architecture for the Graph Convolutional Neural Networks [6], and calculated the degree distribution. Our work and results have been mentioned in the previous sections and the finals results will be incorporated here, in the final report.

6 Critical Analysis

One known complication is the plausibility of Graph Convolutional Neural Networks in this project. GCN [6] approach might have been way too horsepower for this kind of dataset, we chose to go with it in order to understand how they actually work and how can their power be leveraged. As can be seen from the above notes, GCN [6] requires both the adjacency matrix and a feature vector for each node or vertex of the graph in order to create an embedding. But, the dataset we are using does not have such data available, limiting its effect.

References

[1] Paulo Shakarian, Hansheng Lei, and Roy Lindelauf. Power grid defense against malicious cascading failure. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 813–820. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[2] Adilson E Motter. Cascade control and defense in complex networks. *Physical Review Letters*, 93(9):098701, 2004.

[3] Lucas Cuadra, Sancho Salcedo-Sanz, Javier Del Ser, Silvia Jiménez-Fernández, and Zong Geem. A critical review of robustness in power grids using complex networks concepts. *Energies*, 8(9):9211–9265, 2015.

[4] Giuliano Andrea Pagani and Marco Aiello. The power grid as a complex network: a survey. *Physica A: Statistical Mechanics and its Applications*, 392(11):2688–2700, 2013.

[5] Massoud Amin. Challenges in reliability, security, efficiency, and resilience of energy infrastructure: Toward smart self-healing electric power grid. In *2008 IEEE Power and energy*

society general meeting-conversion and delivery of electrical energy in the 21st century, pages 1–5. IEEE, 2008.

[6] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[7] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[8] Michaëll Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering nips. 2016.

[9] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[10] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[11] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[12] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234. ACM, 2016.

[13] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.

[14] Alessandro Vespignani. Complex networks: The fragility of interdependency. *Nature*, 464(7291):984, 2010.

[15] Slaven Peles. Gridkit. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2016.

[16] Jrme Kunegis. KONECT – The Koblenz Network Collection. In *Proc. Int. Conf. on World Wide Web Companion*, pages 1343–1350, 2013.

[17] Us power grid network dataset – KONECT, October 2017.