Network Dismantling Using Reinforcement Learning

Kushal R. Bhandari

1 Introduction

The nontrivial topology and intricate dynamics of complex systems allow certain critical nodes to bear importance among other nodes for network robustness[9]. The effects of targeted attacks on such critical nodes are significant enough to make irreversible changes in the system. Finding such critical nodes is crucial to make, or avoid irreversible changes in the system depending upon its applications. For example, a targeted attack on certain servers could collapse an entire network, perturbations in the food web network could make an irreversible impact, or finding key drug dealers to dismantle a drug network. Such critical nodes in a complex network can help quantify the fragility of a complex network and its susceptibility to attacks.

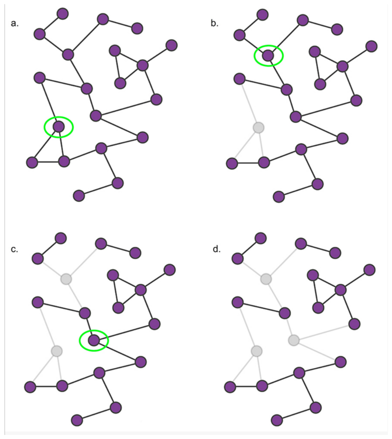

Figure 1: The impact of Node removal [1]

Figure 1 showcases the impact of node removal on the whole network. The largest connected components of the network decrease as more nodes are removed, eventually having isolated components and a completely dismantled network.

However, finding such critical nodes in a complex network is an NP-Hard problem[7]. In this paper and code[1], we used a machine learning model to approximately identify the optimal set of nodes such that its removal can efficiently collapse a complex network.

2 Background

The recent development of machine learning models has become optimal approximation algorithms for researchers to tackle NP-Hard problems. Recently, the state-of-the-art model that utilizes a machine learning approach to optimally dismantle complex networks utilizes graph neural network and reinforcement learning models [4][5]. Fan et al. trained Deep-Q Network model, with node2vec inductive graph representation learning as a Q value approximator, on a synthetic network of type Barabasi Albert Model and size 30-50 for DQN model [4]. Whereas, Grassia et al. used Graph Attention

[1]https://github.com/KBhandari11/NetworkDismantling

Network on a synthetic network to find optimal nodes for network dismantling [5].

2.1 Graph Attention Network(GAT)

Utilizing the two approaches, we implemented Graph Attention Network(GAT) as Q value approximator for DQN model to train an attacker agent to systematically dismantle complex real-world networks. Unlike the standard GAT model, we used GATv2 operator that fixes the static attention problem. First of all, the standard GAT model is a graph neural network architecture that utilizes masked self-attention to regular graph convolution [10]. The standard GAT model has a static attention mechanism since the importance of the attention coefficients is shared across all nodes in the graph, and is unconditioned on the query node [2]. The GATv2 solves this issue through a universal approximator attention function which makes the attention mechanism dynamic [2]. The GATv2 operation is defined as:

$$\mathbf{x}'_i = \alpha_{i,i}\Theta\mathbf{x}_i + \sum_{j\in\mathcal{N}(i)} \alpha_{i,j}\Theta\mathbf{x}_j, \tag{1}$$

$$\alpha_{i,j} = \frac{\exp\big(\mathbf{a}^\top \mathrm{LeakyReLU}(\Theta[\mathbf{x}_i][\mathbf{x}_j])\big)}{\sum_{k\in\mathcal{N}(i)\cup\{i\}} \exp\big(\mathbf{a}^\top \mathrm{LeakyReLU}(\Theta[\mathbf{x}_i][\mathbf{x}_k])\big)}, \tag{2}$$

where, Θ is the learnable parameter and $\alpha_{i,j}$ is the learnable attention coefficient.

2.2 Deep Q-Network(DQN)

Deep Q-Network(DQN) is a reinforcement learning framework that trains an agent utilizing Q-learning as a policy function to maximize the discounted cumulative reward[8]. The Q function obeys the Bellman equation given by,

$$Q^\pi(s,a) = r + \gamma Q^\pi(s', \pi(s')), \tag{3}$$

where, Q^π is the Q function for some policy π, that outputs a value based on the reward r by taking an action a on the given state s. Since we do not know the optimal Q, we use a neural network as the function approximator. In this project, we used GATv2 as the approximator for the optimal Q-function.

3 Methodology

3.1 Training

3.1.1 Game Environment:

For each training episode, the training game environment uses a synthetic graph,$G(V,E)$, of randomly chosen node size of 30-50 and a randomly chosen type of Erdos Renyi, Power Law cluster, Connected

Watts Strogatz, or Barabasi Albert. Initially, all nodes are in their 'active' state, v_n['active'] = 1, viable for an attacker to attack. For each iteration, the attacker attacks a not to change the 'active' state of the node to v_n['active'] = 0. The features used for each 'active' node are degree centrality, clustering coefficient, core number, eigen centrality, and PageRank. The features provide topological information and are computationally less taxing.

3.1.2 State:

During each transition, the attacked node gets isolated. Since, the topology of the graph changes, the features for active nodes are recomputed, and 0 is set as a feature values for attacked nodes.
Each state representation is denoted by the new graph formed by isolating the attack node. The Graph Attention Network uses the recomputed feature and the state of the new graph, to compute Q values of all nodes. Based on the Q values of the legal actions, the agent selects the node with the largest Q value.

3.1.3 Action Sequence:

The action sequence $\Big(a \in \mathcal{A}(\mathcal{S})\Big)$, is defined as the possible set of actions that an agent can take during the given state. The game is designed in such a way that, for any state, the action sequence to the attacker is all the nodes with "active" attributes of 1.

3.1.4 Transition:

A transition is defined as the change from one state to the next state due to action selection. A transition is represented as:

$$\tau(s, a, s', r) : s \rightarrow s', \forall(s, s' \in \mathcal{S}), a \in \mathcal{A}$$

3.1.5 Reward:

The reward system utilizes two different values as incentives, decrease in the largest connected component(LCC), and decrease in the robustness threshold from the previous state. The ultimate objective of the project is to maximize the decrease in LCC value however, for a dense network the decrease in LCC value does not properly incentivize the model [1]. Hence, to resolve such issue, the robustness threshold is introduced to the reward function.
Given, $\alpha = (1 - \frac{2m}{v(v-1)})$, where m is the number of edges and v is the number of nodes; $\gamma_n = |\max\{\delta_k; C_k \in G\}|$, where, C is the connected component in G during state n; and, $\beta_n = \frac{\langle k^2 \rangle}{\langle k \rangle}$, where $\langle k \rangle$, and, $\langle k^2 \rangle$, are the first moment of degree and second moment of degree for graph G during state n.

$$\text{Reward function } = \left(\frac{v'}{v}\right)\left[\alpha * \left(\frac{\gamma_{n-1} - \gamma_n}{\gamma_{n-1}}\right) + (1 - \alpha) * \left(\frac{\beta_{n-1} - \beta_n}{\beta_{n-1}}\right)\right]$$

The reward function is designed such that the α acts as a weighted parameter based on the sparsity of the training graph, and, fraction of alive nodes to all the nodes behaves as a cost for any additional iteration in an episode.

3.1.6 Reset:

An episode of the game lasts until there are no possible sets of legal actions, or the largest connected component of the model is less than 0.1. After an episode, the game resets the graph with a new random graph of node 30-50 and a random graph type.

3.2 Testing

The model was implemented on real world complex networks for evaluation using the same environment as the training.

- Inf-USAir 97 Dataset: The dataset is the category of Infrastructure Networks with 332 nodes and 2,126 edges [3][5].
- Moreno-Crime Dataset: The dataset consists of the Largest Connected Component of the projection of moreno-crime with 754 nodes and 2,127 edges[6].
- Food Web Baywet: The dataset represents food cycle in the cypress wetlands of South Florida during the wet season with 128 nodes and 2,106 edges [6].

4 Evaluation and Analysis

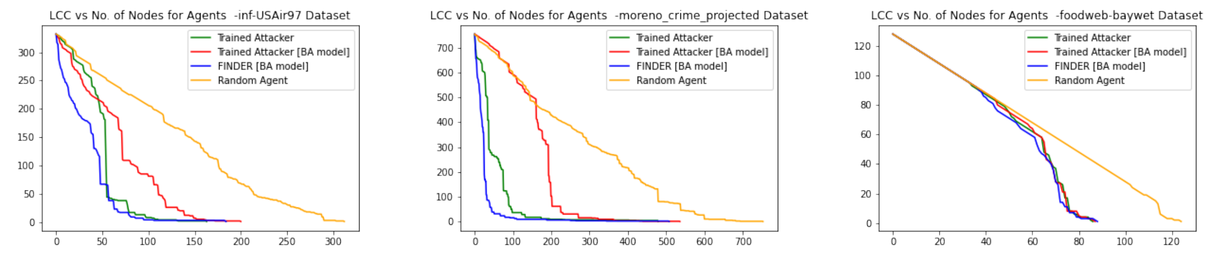

Figure 2: LCC vs No. of Attack Nodes on a real world complex networks

The evaluation of the trained agent on a real-world complex network follows the same environment set up as the training. For a baseline comparison, we utilize two additional agents, FINDER agents based on the state-of-the-art model by Fan et al. and a random agent. Essentially, the FINDER agent is the best-case scenario whereas, the random agent is the worst-case scenario for our model. In addition, we trained another agent from our training setup but with no variation in graph type for the synthetic dataset; we set the graph type to only the BA model.

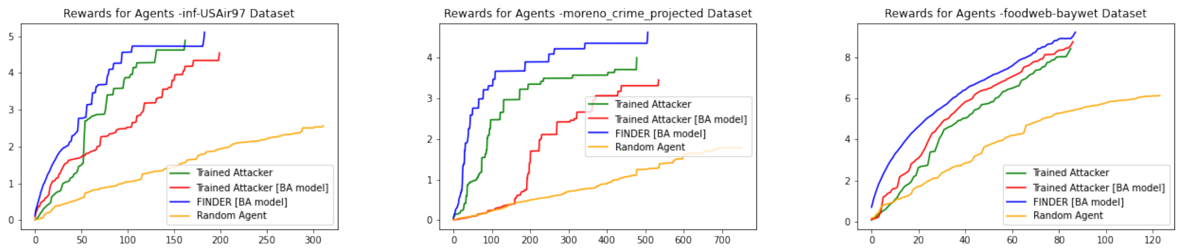

Figure 3: Cumulative reward vs No. of Attack Nodes on a real world complex networks

From figure 2, we can see that our model performs relatively well in different types of real-world complex networks. The difference in the pattern of the plot is due to the different topologies of the network. For the Moreno-Crime-Projected dataset, there is a sudden collapse of the network in comparison to other networks. In the Foodweb-Baywet dataset, for an initial sequences of node removal, there is a linear decrease in the LCC value for sequential node removal. Such linear decrease is because, for a denser network, the network is resilient to any significant damage. However, after some interval, the model collapses quickly. The random agent, that randomly removes a node does not create any significant damage to the network, thus, it suggests that there exist critical nodes in a network that are important for the robustness of the network. One core difference between the agent trained on different type of graphs and same type of graphs is that the 'Trained Agent [BA model]' performs relatively poorly with the Inf-USAir97 network and Moreno-Crime-Projected network, however, with the Foodweb-Baywet dataset, it performs comparably similar to other agents. The synthetic BA model used for the training dataset is homogenous and has relatively higher density, in comparison to other models. This suggests that the agent trained on a similar type of network performs better for a specific type of graph.

Similarly, figure 3 showcases that, the reward function correctly depicts our objective to identify optimal nodes for the successful dismantling of networks. As, the state-of-the-art model, the FINDER agent performs a little better than our model and has a relatively higher reward function.
Hence, the plot suggests that the proposed model still requires more tuning to perform better than the state-of-the-art model.

5 Conclusion

The results from empirical experiments suggest there exists critical nodes in a real-world complex network that are crucial components for the robustness of a network. Using the machine learning approach, it is possible to universally approximate a model that can identify optimal nodes for network dismantling. However, the current approach of using Graph Attention Network and DQN still suffers to achieve the best possible result. In the future, in addition to using other features, we can redesign an environment for a multi-agent reinforcement learning model, and analyze the performance of the cooperative and adversarial dynamics of the two agents. We can create a two-player zero-sum game in a perfect information game setting and use the policy as the strategy for convergence to Nash Equilibrium. The problem will be interesting to study in multiple branches of computer science which include machine learning, network science, and game theory.

References

[1] Albert-László Barabási and Márton Pósfai. *Network science*. Cambridge University Press, Cambridge, 2016.

[2] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *CoRR*, abs/2105.14491, 2021.

[3] Vittoria Colizza, Romualdo Pastor-Satorras, and Alessandro Vespignani. Reaction–diffusion processes and metapopulation models in heterogeneousnbsp;networks. *Nature Physics*, 3(4):276–282, 2007.

[4] Changjun Fan, Li Zeng, Yizhou Sun, and Yang-Yu Liu. Finding key players in complex networks through deep reinforcement learning. *Nature Machine Intelligence*, 2(6):317–324, 2020.

[5] Marco Grassia, Manlio De Domenico, and Giuseppe Mangioni. Machine learning dismantling and early-warning signals of disintegration in complex systems. *Nature Communications*, 12(1):5190, Aug 2021.

[6] Jérôme Kunegis. Konect: The koblenz network collection. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13 Companion, page 1343–1350, New York, NY, USA, 2013. Association for Computing Machinery.

[7] Mohammed Lalou, Mohammed Amin Tahraoui, and Hamamache Kheddouci. The critical node detection problem in networks: A survey. *Computer Science Review*, 28:92–117, 2018.

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, and et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[9] Flaviano Morone and Hernán A. Makse. Influence maximization in complex networks through optimal percolation. *Nature*, 524(7563):65–68, Aug 2015.

[10] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2017.