**EE144: Pandemaniac Report**

**“Engineers at Network”**

*Akshta Athawale, Mannat Singh and Miguel Aroca-Ouellette*

**1. Overview**

For the Pandemaniac competition, our team took a multi-faceted approach wherein we designed different seed generation algorithms for the different portions of the mini-project. The first seed generation algorithm was aimed at scoring well in general, the second and third on beating the TAs and the fourth for scoring well against only other players and TA\_more.

We have three members in our team, Akshta Athawale, Mannat Singh and Miguel Aroca-Ouellette. Akshta was in charge of developing the “Betweenness” algorithm and improving the “High-Degree” algorithm. Mannat handled implementation of the “Clustering” algorithm. Miguel wrote the base class, network simulation wrapper, network visualization, game tree and the “DegreeKiller” algorithm. However, there was significant collaboration on most tasks from every member of the team, particularly in the discussion, design and analysis of algorithms and their results.

**1. High-Degree**

The High-Degree algorithm was our first approach at generating seeds for the provided networks.

**2. Betweenness**

**3. DegreeKiller**

**4. Clustering**

We realised that using just degree and betweenness centralities led us to pick seed nodes in clusters which were really dense. If we were just competing with one team, these approaches worked well. But when there were multiple players, or when we were playing against TA\_more, due to the fact that high degree seed nodes were picked with some team with a very high probability, we would some times end up with very few seed nodes. To address this, we tried to utilise clustering.

The first algorithm we tried was K-means, but it takes a feature matrix as input, not an affinity (adjacency) matrix – this led to high runtimes if we used the adjacency matrix simply as a feature matrix. Since spectral clustering takes adjacency matrices as inputs, and our graphs weren’t too large, it didn’t take too much time to run, and we settled on using spectral clustering. Choosing the number of clusters was difficult, but we ended up choosing 5, as it gave us good results which we could interpret while looking at the graphs visually.

Clustering gave us tags which told which cluster each node was part of. After looking at the clusters, we realised that quite a few times the high degree nodes were all just part of one cluster, which would very often not be the largest cluster. Also, there would be one cluster which would be quite large, which was a collection of nodes which had very small degrees – they couldn’t be part of any cluster, so they were part of this one. We didn’t seed nodes from this cluster.

We tried various approaches to determine the seed nodes once we knew the clustering. One approach was to try and select high degree nodes from each cluster – this worked very poorly, as we weren’t able to “take over” even one cluster in reasonable time, and in a few iterations all our nodes would get eliminated.

We then focussed on ensuring that we spread our epidemic on only a few clusters. We used the average degree in a cluster to determine its density, and the number of nodes inside it to determine its size. After trying various combinations, we ended up with the following criterion –

* Cluster the graph into 5 clusters
* Find the two densest clusters, based on the average degree
* Choose the cluster with the highest number of nodes, out of the two densest clusters

After this, we again randomised selecting the highest degree nodes, but only from that cluster. Another improvement we applied was to use DegreeKiller, but with our candidate nodes restricted to be in that specific cluster.

With this, we were able to comprehensively beat TA\_more for all the days.

We tested this approach in round 5, where out of the 4 graphs with 8 players, we got 20 points for two graphs, 6 points for one, and 1 point for the last.

**5. Game Tree and Seed Tournament**

**6. Visualization and Testing**

**7. Conclusion**

**8. References**