CS144 Miniproject: Pandemaniac

Wen Gu, Sha Sha, Yu Wu

1 Centrality Measures

The key to this problem is to find the nodes which could help spread epidemic to more nodes. We studied three different centralities to measure the importance of nodes when spreading epidemic.

1.1 Degree Centrality

Degree centrality is defined as the degree of one node. Intuitively, a node with higher degree means that the epidemic could be spread to more nodes at one step.

Let d_i denote the degree of node *i*. The degree centrality of node *i* is defined by

$$C_D(i) = \frac{d_i}{n-1}$$

The degree centrality is proportional to the degree. To simplify the calculation, we use degree as measure of importance. The complexity of computing degree is $O(E)$, where E is the number of edges in the graph. Since the input is adjacent list, we calculated degree by counting edges connected to each node as its degree.

1.2 Closeness Centrality

Closeness centrality is defined as the reciprocal of the average distance between node *i* and other nodes. Let $l(i,j)$ denote the distance (length of the shortest path) between bodes *i* and *j*. The closeness centrality of node *i* can be calculated by

$$C_C(i) = \frac{n-1}{\sum_{i \neq j} l(i,j)}$$

The node with higher closeness centrality means its average path to other nodes is shorter, which implies that with same steps, on average, nodes can be reached from the nodes with higher closeness centrality with fewer steps. However, the computation of closeness centrality is complex since all the shortest path need to be found. When applied Dijkstra Algorithm to find the shortest path starting from node *i*, the time complexity to is $O(ElogV)$, where E is the number of edges in the graph and V is the number of nodes in the graph. Therefore, the total time complexity to compute

closeness centrality for the graph is $O(VElogV)$. We use closeness_centrality function in networkx package to compute closeness centrality.

1.3 Betweenness Centrality

Betweenness centrality is defined as the number of those shortest paths that pass through node *i*. Betweenness centrality measures the importance of a node with respect to connecting other nodes in the graph. Let $P(j,k)$ denote the total number of shortest paths between nodes, and $P_i(j,k)$ denote the number of those shortest paths that pass through node *i*. The betweenness centrality of node *i* can be computed by

$$C_B(i) = \frac{\sum_{j,k:j\neq k,j,k\neq i} \frac{P_i(j,k)}{P(j,k)}}{\binom{n-1}{2}}$$

We use betweenness_centrality function in networkx package to compute betweenness centrality. The time complexity of the algorithm to compute betweenness centrality in unweighted graph is $O(VE)$.[1]

1.4 Mixed Strategy

After testing on several graphs, we found that there is a lot of overlap among top N nodes with highest degree, highest closeness centrality, or highest betweenness centrality. In addition, if more than one team pick the node, then none of the team gets that node. Therefore, we need to find a trade-off between choosing nodes with higher importance and avoid being the same with others' choice. We want to randomly choose nodes in the union set of top N nodes with these three centralities.

Denote m as the number of seeds we need to choose finally. Denote k as the ratio of single set, which means we pick top km nodes from each centrality. Denote D, C, B as the set of top km nodes with highest degree, closeness centrality and betweenness centrality respectively. The fig 1 below is Venn diagram for the three sets. We want to select a percent of final nodes from part 1, which means these nodes are of great importance in spreading epidemics, but they are also quite popular be to selected by other teams. The rest 100-a percent of final node are selected from the union of part 2,

3, 4, 5, 6, and 7, which are less important but also be less popular.

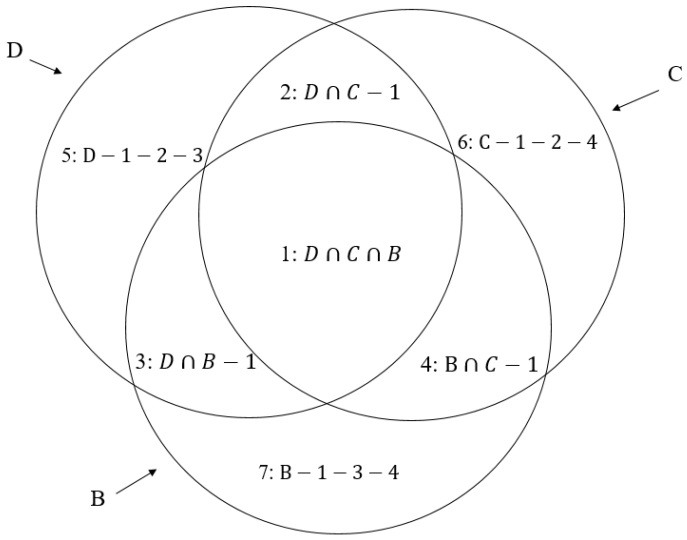

fig 1 Venn diagram for three sets

The specific values of k and a depend on the graph structure and number of players. A larger k or a smaller a means we could tolerate lower value of centrality.

2 Clustering Method

When adopting Mixed Strategy discussed above, as it involves calculations about degree, closeness and betweenness, we observe that it can be too slow on big graphs to satisfy the time limit of 5 minutes. Thus, clustering is important and necessary as the calculations can be reduced by cutting the big graph into several relatively small clusters, we can focus on one of the clusters and select nodes from this cluster. In this way, we can reduce the graph complexity and focus our nodes on this cluster so as to occupy it. On the graph, we pay more attention to the connectivity instead of compactness, therefore, we apply the spectral clustering method.

2.1 Graph Laplacian

The spectral graph theory arises from the question about what properties of a graph are exposed or revealed if we represent the graph as a matrix and study the eigenvectors and eigenvalues of that matrix.

Given a graph with n vertices, its Laplacian matrix $L_{n\times n}$ is defined as:

$$L = D - A$$

Where D is the diagonal matrix with $D_{ii} = d_i$, the degree of node i, and $D_{ij} = 0$ for $i \neq j$, A is the adjacency matrix with $a_{ij} = 1$ if edge $i \to j$ exists and otherwise $a_{ij} = 0$.

Therefore, the elements of L are given by:

$$L_{ij} = \begin{cases} \deg(\mathrm{v_i}), & \text{if } \mathrm{i = j} \\ -1, & \text{if } \mathrm{i \neq j \text{ and } v_i \text{ is adjacent to } v_j} \\ 0, & \text{otherwise} \end{cases}$$

We can observe that L is real-valued, symmetric and positive semi-definite. It has n non-negative, real-valued eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ and orthogonal eigenvectors $v_1, v_2, \ldots, v_n$. The eigenvalues of L contain information about the graph structure.

The Laplacian always has at least one eigenvalue that is 0 and the number of zero eigenvalues equals the number of connected components of the graph. The eigenvector corresponding to the smallest non-zero eigenvalue will be minimizing the sum of squared distances, which is good for partitioning and settle for clusters.

2.2 Spectral Embedding

Spectral embedding is for non-linear dimensionality reduction. We use sklearn.manifold.SpetralEmbedding package to generate the spectral distribution and reduce the dimension to 2-D. In this part, it forms an affinity matrix given by the specified function and applies spectral decomposition to the corresponding graph Laplacian. The resulting transformation is given by the value of the eigenvectors for each data point.

Here, we interpret the adjacency matrix A as precomputed affinity matrix and set the projected subspace of two dimension. In this way, we find an embedding of the graph into 2-D and the spectral distribution reveals its structure, i.e., the points connected on the graph stay as close as possible after embedding. Base on the spectral distribution, we can observe the clustering information and further cut the big graph into clusters. At this time, the $A_{embedding}$ is now in the shape of $n_{samples} \times n_{component} = n_{samples} \times 2$.

2.3 Spectral Clustering

To do the spectral clustering, we firstly find the largest connected component in the graph by using networkx. connected_component_subgraphs and take it as our origin graph, which helps discard isolate and useless nodes. We then use sklearn.cluster.SpectralClustering package to cut the graph into n_clusters clusters. As in practice, spectral clustering is very useful when the structure of the individual clusters is highly non-convex or more generally when the measure of the center and spread of the cluster is not a suitable description of the complete cluster.

We firstly try to interpret A as precomputed affinity matrix and do clustering directly from adjacency matrix A, but the clustering performance is not as good as we expect. The reason may because that the big graphs are too complex. Thus, we change the mind to firstly do the spectral embedding and project the data points into 2-D subspace, and then partition on the embedding $A_{embedding}$ as now the spatial distribution of the points can represent the connectivity. We choose the affinity as "nearest_neighbors"

that produces similarity scores. Afterwards, we get the cluster labels for each data point. The number of clusters we use can vary on different graphs. We observe the clustering properties on both the graph drawn by networkx.draw and the spectral distribution via embedding.

For small graphs with no obvious clustering, we don't need to do the clustering as it just increases the complexity of the algorithm. While for those graphs which have obvious clustering, we should do the clustering not only for computational efficiency but also because selecting all our nodes from one cluster could help us to occupy the cluster with higher probability. Thus, we set a cluster flag = 1 if we want to cut the graph into clusters, and then select the largest cluster as our graph for centrality measurement calculating. Besides, we limit the largest number of nodes that the final cluster we choose can possess so as to limit the time for calculation.

3 Game Theoretic Principles

With the rule that the seed node selected will be canceled out if the node is selected by two or more teams, the game theory would be applied to this project to help with seed nodes selection process. The key is that we not only need to maximize the number of nodes but also concern the seed nodes collision problem with the other teams in case the seed nodes are canceled out.

3.1 1 vs 1 competition strategy

Several of the practice graphs require 2 teams to compete with each other: one scenario is the competition with other teams and another scenario is the competition with TA. In 1 vs 1 battle game, we found that the team, which selected more nodes in the highest ranked nodes in centrality measure, would always win the game.

In 1 vs 1 game, we have tried to avoid collision by randomly selecting several nodes which are not in the highest ranked node. If 10 nodes were required, then we tried to select the several nodes in the top 10, as well as some nodes ranked from top 10 to top 20. In this way, we kept the possibility that we have the advantage over another team in case they abandoned the top 10 seed nodes to avoid collision, along with the hope that we still have some seed nodes ranked from top 10 to top 20 left once nodes in highest-ranked cancel out due to collision. However, this strategy was not as good as we thought because the game is symmetric.

In a single node selection, for only infection maximization purpose, we can make an assumption that seed nodes with higher rank would have better infection effect than the lower ranked seed nodes in centrality measure. Assume node i, denoted by Vi , is the highest ranked node. In addition, we have team A and team B compete with each other to choose node Vi. If either of team A or team B chooses node Vi, the payoff is 1 for the one who gets the node, and 0 for the one who does not get the node. If both teams choose node Vi, then the payoff for both team would be 0 due to cancel out. Then we have the payoff table as follows.

Table 1 Payoff Table for Single Node Selection In Two Team Competition

	B chooses V_i	B does not choose V_i
A chooses V_i	0,0	1,0
A does not choose the V_i	0,1	0,0

As we observed, the table is perfectly symmetric. For team A, choosing Vi is better than abandoning Vi in the sense that choosing Vi still keeps the chance to get the Vi, and abandoning Vi would leave team A nothing. The same reasoning also holds for team B. Thus, the dominant strategy for both teams is choosing the highest ranked node Vi.

In addition, the game is still symmetric when choosing a set of seed nodes rather than a single node. Denote strategy where a team chooses both high ranked nodes and low ranked nodes above as M. Denote strategy where a team chooses only high ranked nodes as H. Both team A and team B would choose either strategy M or strategy M. In the case both A and B chooses M, it is hard to say which team would win, and thus the payoff for both team would be 0.5. If one of the team chooses H and another team chooses M, then the team, who chooses H, would have the payoff as 1 and another would have the payoff as 0. The reason is that the number of nodes canceled would be the same for both teams, and the team with higher ranked surviving nodes would win. If both teams choose strategy H, then it would be more likely a random result and both teams get payoff 0.5, because different centrality measures would rank the importance of the nodes differently. Then we have the payoff table as follows.

Table 2 Payoff Table for Multiple Seed Nodes Selection In Two Team Competition

	B chooses H	B chooses M
A chooses H	0.5,0.5	1,0
A choose M	0,1	0.5, 0.5

As we observed, the table is still perfectly symmetric. For team A, choosing strategy H is better than strategy M in the sense that choosing H always have a better payoff. The

same analysis also holds for team B. Thus, the dominant strategy for both team is choosing a set of nodes which rank highest in centrality measure.

According to the above game theory analysis, we should select the highest ranked nodes rather than avoiding collision. What's more, the analysis also gives us the intuitive reason why collision avoidance algorithm is not as good as choosing the highest-ranked node directly. Hence, in the later 1 vs 1 competition, we change our strategy to choose the seed nodes with the highest-rank in the mixed strategy of centrality measure.

3.2 Multiple Teams Competition Strategy

Another part of the Pandemaniac is the competition between multiple teams. It is much harder to analyze than 1 vs 1 competition since it is a greater system of multiple players and multiple strategies. Each team might take the different strategy and small change in strategy might take the result to a different direction. However, the two main key ideas do not change too much from the 1 vs 1 strategy: one is for maximizing the number of infection and another is for collision avoidance.

3.2.1 Degree Centrality

At first, we tried to use the highest degree centrality. The result shows that the highest degree measure does not work very well due to the number of teams who apply the same strategy. A large number of teams who also used the highest degree leads to the huge amount of cancellation of the seed nodes, and thus our seed nodes did not survive from cancellation. What's more, even if the seed node selected by highest degree survived, we cannot guarantee that the seed nodes can infect the neighbor nodes since the distances between the seed nodes are great. The highest degree algorithm cannot guarantee that the seed nodes are close to each other enough to infect the neighbor nodes. As result, the highest degree is not an efficient algorithm.

3.2.2 Closeness Centrality/ Betweenness Centrality

We also tried the closeness centrality and the betweenness centrality algorithm. The result shows that these two algorithms work better than the degree centrality in the sense that they somehow guarantee that the selected seeds nodes are close to each other and the neighbor nodes can be infected efficiently. However, there is still a lot of teams

using either of these two algorithms, sand thus directly applying these two algorithms would be problematic since cancellation happens as selected seed nodes are the same. What's more, we also found that the ranking generated by closeness centrality is similar to the ranking generated by betweenness centrality. This would also lead to collision problem when two teams choose the highest-ranked node from either of these two algorithms.

3.2.3 Mixed Centralities Strategy

To avoid the collision problem mentioned above, we tried to mix degree centrality, closeness centrality, and betweenness centrality. At each iteration, we randomly selected several nodes from the overlap part of three rankings, as well as several nodes from the non-overlap part of three rankings. The overlapping part, as the most important nodes selected by all three algorithms, gives the opportunity to capture the center of the graph if other teams tried to avoid the collision by giving up these nodes. The non-overlap part allows part of seed nodes to survive when those important nodes are also selected by other teams. The randomness also ensures the avoidance of collision to combat the constant strategy. As mention in section 1.4, several hyper-parameters were adjusted for type/ size of graphs and number of teams. What's more, we also used the clustering to reduce the graph and boost the computation speed. The result turns out that this method works well on the large graph.

From our experience, the collision avoidance is much more important than the maximizing the infection because a large number of teams aim at top ranking nodes. Once survived from the collision, then the seed nodes can start to infect others.

4 Strategy Summary

4.1 Strategies to Beat TAs

- Beat TA fewer

  We use simple top 10 nodes with highest closeness degree to beat TA since they have less seeds.

- Beat TA degree

  We know that TA use highest degree as measure of importance of nodes. Therefore, we cannot use degree centrality to choose our seeds, which will lead to cancellation. We choose top 10 nodes with highest closeness degree to beat TA.

4.2 Strategies in Multi-Players Game

Our strategy mainly contains two steps: 1) split the graph into clusters; 2) use mixed strategy to select seeds.

When the size of graph is very small or we find that the graph only contains one cluster after visualization, we use the original graph to pick seeds. Otherwise, we split the graph into several clusters and then choose the subgraph with most nodes.

Then we need to decide the strategy to pick the seeds. If the graph is too large to use compute closeness centrality or betweenness centrality within the prescribed time, we simply use highest degree measure. Otherwise, the mixed strategy is applied. Then we need to choose appropriate k and a. When there are many players in the graph, we want to avoid to choose the same nodes with other teams. Therefore, we use a larger k, such as 1.5 or 2, and a smaller a, such as 0.4. If there are less players, we use a smaller k, such as 1 or 1.5, and a larger k, such as 0.5 or 0.6.

With such strategy, we finally got the fourth place in the tournament.

5 Suggestions

1. The workload of the homework is somehow heavy. The number of homework can be reduced, and we would like to take more time into the interesting hands-on project like Pandemaniac and Rankmaniac.
2. The practice and competition phase of Pandemaniac last a full week in total, which is somehow too long, and it makes all three of us feel tired. It would be better if the practice stage is reduced from 5 days to 3 days and competition stage from 3 days to 2 days.
3. It will be interesting if 1 vs 1 game is also added to the competition phase of Pandemaniac. In addition, we can not only have a scoreboard of competition between multiple teams, but also a scoreboard for 1 vs 1 competition.

6 Team Information

6.1 Team Name

_SWG

6.2 Team Member

Wen Gu

Sha Sha

Yu Wu

6.3 Divison of Labour

Wen Gu: Collaborate on the basic strategies. Implement the Mixed Strategy. Collaborate on the design of Clustering Method. Analyze the graph structure.

Sha Sha: Implement the basic strategies of highest degree, closeness and betweenness. Collaborate on the design of Mixed Strategy. Collaborate on the implementation of Clustering Method. Analyze other competitors' strategies.

Yu Wu: Implement the basic strategies of highest degree. Design the Mixed Strategy. Collaborate on the design of Clustering Method and implementations. Analyze the graph structure. Build up and maintain the code structure. Strategy submission.

6.4 GitHub

https://github.com/wuyudd/CS144_Pandemaniac

Reference

[1] Brandes, U. (2001). A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, *25*(2), 163-177.