Caltech CS 144, Winter 2015
Pandemaniac Report
Jan Van Bruggen and Sean Dolan (2pox)

Our Approach

We began by considering the victory condition: the victor is chosen (and the rest are ranked) by the total number of nodes of their color (controlled) at the end (stable or 50 iterations). The simplest approach is that choosing the highest-degree nodes gives us the most nodes on the initial iteration. However, this might only achieve either limited influence in many parts of one cluster or limited growth beyond a densely-packed cluster.

To figure out which nodes in a given graph are important we turned to HW 2, where we considered various methods of ranking the importance of nodes. A node with high closeness means that, after the initial seeding, we might quickly grab nodes in many different clusters of the graph, while a node with high betweenness might allow us to control bottlenecks and prevent the growth of other teams' seeds.

We then began considering other teams' seed choices. A collision seems like something to avoid, as it results in neither team getting the node, but we decided to look at it from a game-theoretic approach. Consider the following payoff matrix, which assumes X and Y are 2 nodes that we have correctly ranked (ie we end up with more nodes by seeding X than by seeding Y) and V() is the value function:

	Other Team Chooses X	Other Team Chooses Y
We Choose X	0, 0	V(X), V(Y)
We Choose Y	V(Y), V(X)	0, 0

There are then 3 distinct possibilities for our change in utility. Since this is a direct competition, we can consider their score as being subtracted from our own:

$$\Delta U = 0 - 0 = 0$$
$$\Delta U = V(X) - V(Y) > 0$$
$$\Delta U = V(Y) - V(X) < 0$$

We can see that we have a dominant strategy of always picking the higher-ranked node X. However, this completely falls apart when considering games of more than 2 teams, where unless all teams collide on all nodes, the non-colliding teams gain a strict node advantage.Therefore we decided that in games with more than two players, collisions should be purposefully and strongly avoided.

We then considered weighting each centrality differently, but we decided against this. As the TA teams don't compete in the tournament other teams' algorithms changed significantly throughout the project, the only data that would help prepare for the tournament was the data from the tournament and perhaps the data from the two days beforehand. After

analyzing this data, we did not see any convincing reason to weight one centrality higher than another.

Another (fairly cheap) computation to glean data from nodes is to get their clustering. High clustering seems like it would be good for a seed node, as it would be able to secure its starting position However, once clustering was added to our algorithm, the seeding algorithm began taking much longer than 5 minutes to run for some graphs, so we began to look at ways to computationally simplify the approach. We began by analyzing the centralities and clustering coefficients of other teams' seed nodes, in order to judge what worked and what didn't, and to see if we could glean any strategies to counter theirs. We found that most of the high-scoring teams picked low-clustering nodes and that they did not strongly select for high betweenness centrality.

Using this information, we removed clustering from our calculation. A seed node with high clustering would have low degree centrality and only spread its influence to nodes in its immediate neighborhood, resulting in slow growth. Betweenness centrality was removed it became too computationally expensive for the 8 and 16 player graphs. . We attempted to reduce the calculations by only considering the paths between 500 random nodes nodes (j and k in the definition given in HW 2), but this reduction to about 1/20 of the graph might begin to skew the betweenness values for some graphs.

Our Algorithm

First we calculate the closeness and degree centralities for each node. A high degree centrality allows us to convert a large number of nodes quickly, while a high closeness centrality allows our influence to expand to other areas rapidly. Combined, these rankings identify seeds that will allow us to convert uncolored nodes in many areas of the graph. We weight these two centrality measures equally, because our simulation testing was inconclusive regarding which measure contributed more to a strategy's success. Considering the following aspects of our algorithm, we see no convincing reason to prefer one measure over the other.

We then remove the bottom 90% of nodes with regard to the sum of the two measures of centrality, or the "score" of a node. We then draw our seeds from a probability distribution over this top 10% of nodes, where the probability of selecting a node grows linearly with its score. This hedges our seeds against the threat of repeated collisions across rounds by not repeatedly picking the same nodes as another team. We accelerate this computation by considering the highest-scoring nodes first, which greatly reduces the number of iterations required to find a fixed number of nodes by probabilistic means.

Additionally, we never pick a node with a closeness or degree centrality in the top 1% of the graph. This further hedges our seeds against the threat of collisions, as the data gleaned from histograms of prior rounds' seed nodes showed that some groups' strategies heavily weighted one or both of these centrality metrics. Often the highest scoring teams avoided the top 1% most-central nodes (whether this is intentional or a by-product of a radically different algorithm is unknown), so we emulated this in our design.

You can view our source code in `seed.py` at https://github.com/jvanbrug/2pox.

Suggestions

Disseminating how the graph was produced (whether from social network data or generated randomly with a model such as Erdős–Rényi) would allow more analysis regarding possible graph features . For example, if the number of graph components was known, teams could check for components and seed each one (as a current component count may prove too costly to implement given the time limits).

Contributions

Jan Van Bruggen was responsible for the framework of the project and the data analysis, while Sean Dolan was responsible for the seeding algorithm. This report was jointly written.