CS/EE 144 Pandemaniac Report

Team kraftwerk
Fabian Boemer, Jessica Li

February 25, 2016

TODO: parallel computations? Jessica's things (spectral clustering?), visualizations. Descriptions of different centralities and why they wold be promising. Intro/Conclusion

1 Introduction and Overview

2 Visualization

3 Algorithms

3.1 Day 1: Cancellation

The first approach was a naive cancellation approach designed to beat TA-degree. Algorithm 1 gives the pseudocode for the cancellation approach.

Algorithm 1 Cancellation Algorithm

```
procedure CANCELLATION(G, n)
    Order the nodes by degree centrality
    Select the n − 1 nodes with highest degree centrality
    Select the previously-unselected node adjacent to the vertex of highest-degree.
    return selected nodes.
```

Time constraints prevented us from testing this strategy before the Day 1 run. We visualized the results from the Day 1 run against TA-degree, and visualized each iteration in Figure 1.

The top-left plot is iteration 1, where we notice only one node of each color is selected, by construction. The top right right is iteration 2, where we see the red node adjacent to the bottom-left corner, with degree 139. So our strategy successsfully claimed the highest-degree node. However, the TA's blue seed node has high degree, and claimed several blue nodes. Then, at iteration 3, we see the highest-degree node failed to convert many nodes. The TA already has spread to enough nodes that converting any new nodes is difficult. By iteration 4, it is clear the blue nodes have won. Thus, our cancellation approach is too slow; the controlling the highest-degree node at iteration 2 is not enough to win against TA-degree.

As simulated, we did not beat TA-degree on day 1. We did, however, beat TA-fewer. This is not unexpected, since the $n - 1$ nodes with high degree centrality are a decent selection of important nodes.

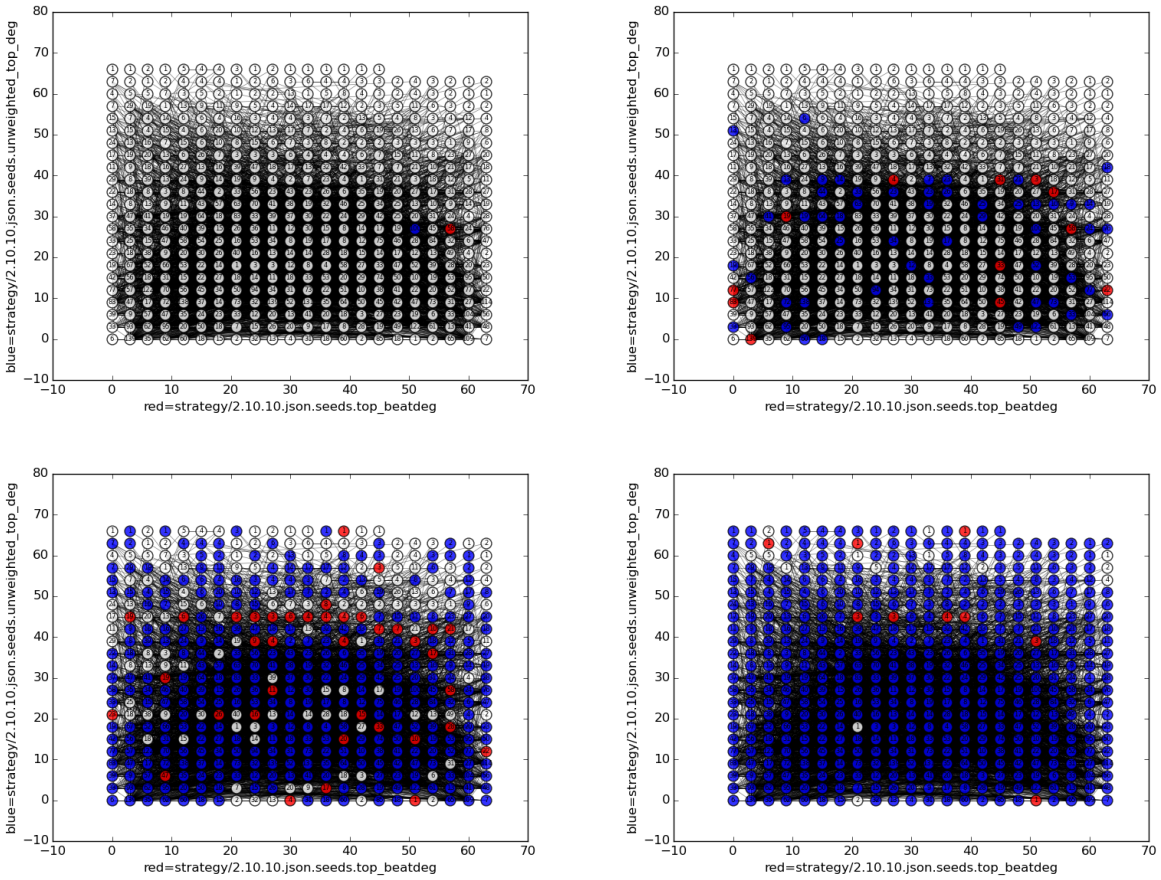

Figure 1: Results from Day 1 against TA-degree. Our team's nodes are red; the TA's nodes are blue. The numbers in the nodes indicate the node degree.

3.2 Day 2: Centrality Measures

On day 2, we focused on developing a strategy to beat TA-degree. We explored several different measures of centrality, including betweeness, current flow betweenness (cfbet), closeness, and eigenvector centrality. First we tried taking top n most-important nodes, as ranked by each centrality measure, excluding degree-centrality since TA-degree uses degree centrality. We modified 'sim.py' to simulate each selction strategy against TA-degree. Table 1 shows the results of our simulations

From the Day 1 simulations, we see closeness and eigenvector centrality consistently outperformed TA-degree. For our submission, we set up a local pipeline to generate seeds for current flow betweeness, closeness, and eigenvector centralities, and simulate TA-degree against each centrality measure. We expeceted at least one of the three strategies to beat TA-degree on graph 2.10.11, based on the performances on graphs 2.10.10, 2.10.20, 2.10.30.

Unfortunately, during submission, we simulated narrow losses against TA-degree with closeness and eigevector centralities, with a heavy loss with current-flow betweenness. The simulation was correct, as we failed to beat TA-degree.

Day	Graph	Centrality measure	Strategy count	TA-degree count	Ratio
1	2.10.10	betweenness	18	482	0.04
	2.10.10	cfbet	491	9	0.99
	2.10.10	closeness	386	113	0.77
	2.10.10	eigenvector	394	106	0.79
	2.10.20	betweenness	43	455	0.08
	2.10.20	cfbet	43	455	0.08
	2.10.20	closeness	452	45	0.90
	2.10.20	eigenvector	429	69	0.86
	2.10.30	closeness	475	22	0.96
	2.10.30	eigenvector	469	28	0.94
2	2.10.11	cfbet	31	468	0.06
	2.10.11	closeness	231	263	0.47
	2.10.11	eigenvector	239	256	0.48
	2.10.21	closeness	277	216	0.56
	2.10.21	eigenvector	206	283	0.42
	2.10.31	closeness	35	456	0.07
	2.10.31	eigenvector	28	464	0.056

Table 1: Summary of local testing results for each centrality measure for Day 2.

We spent some time trying to visualize the structure of the graphs. Unforunately, the visualizations produced by networkx's 'spring layout' were inconsistent between successive drawings, and the remaining visualization formats did not expose the structure of the graph. Thus, we used visualization only sparingly from this point forward.

We tried clustering the grpaph using networkx's, k_components = apxa.k_components(G). For 500-node graphs, the clustering took 129 seconds, which was deemed too long for the 3 minute deadline.

3.3 Day 3: Monte-Carlo

Having failed to beat TA-degree, we increased the scope of our approach. We explored several new measures of centrality, and began exploring blending of results. We included Katz centrality and dispersion centrality, **which are good for what?** Algorithm 2 is the pseudocode for this approach.

This is a brutish approach, which does not distinguish between the efficacy of each centrality, nor the complementary advantages of each centrality measure. For ecample, nodes with high betweenness measure are likely prevent cascades from spreading, while nodes with high closeness centrality spread quickly to all nodes in a cluster. Nevertheless, this approach successfully generated several strategies simulated to win against TA-degree. Having not yet beat TA-more, we varied our final output among the top-10 best-performing strategies simulated to beat TA-more. This provides more variance in the seeding, and makes the seeding more robust against an adverserial TA seeding.

As simulated, we successfully beat TA-degeree with this approach. However, we still failed to beat TA-more.

Algorithm 2 Promising Algorithm

procedure PROMISING(G, n)
    Order the nodes by degree centrality, eigenvector centrality, Katz centrality, closeness centrality, dispersion centrality.
    Initialize dictionary d with keys all the nodes and values as 0
    for each centrality measure **do**
        for node in top n ranking **do**
            Let d[node]$+ = n-$ ranking.
    Sort d descending by value and choose 12 highest-value nodes as 'promising'
    for each of $\binom{12}{10}$ combinations of 10 seed nodes **do**
        Simulate the seed against TA-degree
    Order seeds by number of nodes claimed against TA-degree
    while Final Seeding not complete **do**
        for the top ten seeds ranked by performance **do**
            populate Final Seeding with the ten seeds
return Final Seeding.

3.4 Day 4: Blending

We expanded on our idea of blending promising nodes into a more refined approach. We took the n most promising nodes based on degree centrality, betweenness and eigenvector centrality. We reasoned that nodes with high degree centrality would effectively conquer clusters, while nodes with high betweeness would effectively blockade cascasdes from proceeding between clusters. Up to this point, eigenvector centrality performed quite well and was computed very rapidly, so we decided to include the nodes with high eigenvector centrality as well.

Algorithm 3 Hybrid Algorithm

procedure HYBRID(G, n)
    Order the nodes by degree centrality, eigenvector centrality, Katz centrality, closeness centrality, dispersion centrality.
    Generate the set s of top-n ranked nodes in each centrality measure
    Generate 100 random selections of n nodes from s.
    for each of 100 seed node selections **do**
        Simulate seed nodes against TA-degree with $1.2n$ nodes.
    return best-performing set of seed nodes.

LOCAL RESULTS!?

On the Day 4 run, we still failed to beat TA-more. Thus, the brute-force generation of 'important' nodes failed to quickly cover enough variation in seed nodes to beat TA-more

3.5 Day 5: Clustering

On Day 5, we discovered a faster way of partitioning based on the Louvain method [1], and implemented in networkx. This clustering approach was much quicker than the previous approach using k_components = apxa.k_components(G). The size of the clusters varied from less than 10 to over 100 and were not disjoint. We adapted previous approaches to clustering, described in Algorithm 4.

Algorithm 4 Cluster Algorithm

```
1: procedure HYBRID(G, n)
2:     Cluster G using Louvain method
3:     for each cluster c do
4:         Use promising(c, n) to generate 'most-important' node within the cluster.
5:     for each of 3 populating strategies do
6:         Initialize empty FinalSeeding list
7:         while FinalSeeding length less than n do
8:             for each cluster c do
9:                 Insert most-important node(s) from c not yet in FinalSeeding
10:            Simulate FinalSeeding against TA-more Day 4
11:    return best-performing FinalSeeding
```

In line 8, we explored three methods of choosing the number of most-important node(s) selected. For a given cluster c, we populate FinalSeeding with the number of nodes proportional to one of:

(1) the total degree of nodes in c

(2) the number of edges in c

(3) 1 (each cluster has the same number of nodes)

We simulated the results against the nodes picked by TA-more on previous days. Our findings are summarized in Table 2.

Graph	Strategy	Proportion Nodes Claimed	Games won
2.10.31	(1)	0.04	0
2.10.31	(2)	0.06	1
2.10.31	(3)	0.04	0
2.10.32	(1)	0.60	47
2.10.32	(2)	0.21	4
2.10.32	(3)	0.21	7

Table 2: Summary of local testing results using clustering approach for Day 5.

A visualization **SHOW VISUALIZATION!** between graphs 2.10.31 and 2.10.32 suggested 2.10.32 was more clearly clustered than 2.10.31, suggesting our approach (1) works well for clustered graphs. This makes sense for clusters of roughly equal size. In retrospect, perhaps we were placing

too much weight on small clusters. Rather, perhaps we should have chosen only nodes from the largest cluster.

Under time constraint, we decided to use the number of nodes proportional to the number in each cluster, since this was the only winning approach we discovered. Unfortunately, this failed to beat TA-more.

4 Conclusion

References

[1] Fast unfolding of communities in large networks, Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Renaud Lefebvre, Journal of Statistical Mechanics: Theory and Experiment 2008(10), P10008 (12pp)